Computer programs for texture simulation

Leffers, T.

Publication date:
1973

Document Version
Publisher's PDF, also known as Version of record

Risø Report No. 283

Risø Report No. 283

Danish Atomic Energy Commission

Research Establishment Risø

Computer Programs for Texture Simulation

by Torben Leffers

March 1973

Sales distributors: Jul. Gjellerup, 87, Sølvgade, DK-1307 Copenhagen K, Denmark

Available on exchange from: Library, Danish Atomic Energy Commission, Risø, DK-4000 Roskilde, Denmark

U D C 681.3.06 : 539.374 : 621.771

March 1973 Risø Report No. 283

Computer Programs for Texture Simulation

by

Torben Leffers

Danish Atomic Energy Commission
Research Establishment Risø
Metallurgy Department

Abstract

A description is given of three computer programs (ROLTEX 2, ROL - TEX 1, and HORS dOEUVRE) for the simulation of the plastic deformation in face-centred cubic polycrystals and the derived development of rolling texture. The description includes a variety of test examples demonstrating that the programs work correctly. The report does not deal with the actual results produced by the programs and the physical interpretation of these results.

ISBN 87 550 0189 0

CONTENTS

1. INTRODUCTION

The computer programs ROLTEX 1 and ROLTEX 2 are used for simulating the plastic deformation during rolling of face-centred cubic polycrystals and the derived texture development. HORS dOEUVRE is an auxiliary program to the ROLTEX programs.

The results obtained with the programs and the discussion of the physical significance of these results are published elsewhere[1-6]. This report will provide a detailed description of the organization of the programs, of the mathematical formulation of the problems, and of the testing procedure applied to ensure that the programs actually do perform the desired calculations. Listings of the programs are included as appendices.

ROLTEX 1 was written first, but before it came into real use, it was realized that so many alterations and so much rationalization were needed that it would be the easiest to write a new program, ROLTEX 2. All the results referred to in refs. 1-6 have been obtained with ROLTEX 2, and therefore only ROLTEX 2 will be described in detail. The only reason for including ROLTEX 1 is that some of the tests referred to can only be performed with this program. The parts of the auxiliary and test program HORS dOEUVRE that are different from ROLTEX 2 will also be described.

2. THE COMPUTER PROGRAMS

2.1. Mode of Operation

A predetermined number of crystals are treated one by one. The starting orientation of the crystals may be fed into the computer, or the computer may generate random orientations.

In the individual crystals the program selects the $\{111\}\langle 110\rangle$ slip system with the highest resolved shear stress. The resolved stress is calculated from the following external stresses: (i) the basic stress system which is a tensile stress in the rolling direction combined with a compressive stress in the normal direction, the two stress components being of equal size numerically, (ii) if wanted, additional stresses restricting the changes in shape that the crystals can undergo, and (iii) if wanted, random stresses (the physical significance of these stresses are given in refs. 2, 3, 6).

When the most heavily loaded slip system has been found, a certain amount of slip takes place on this system, and the crystal lattice is rotated

correspondingly. The lattice rotation takes place such as to fulfil the following two requirements: (a) a string of material lying in the rolling direction before slip must retain this orientation after deformation and lattice rotation, and (b) a plate of material lying parallel to the rolling plane before slip must also retain its orientation after deformation and rotation (see refs. 1, 2, 3, 6). The change in shape caused by the slip is registered; it is used for calculating the total deformation and, if the stresses (ii) mentioned above are applied, for calculating the new shape-restoring stresses. Then a new slip event with rotation takes place like the one already described, and so on until the predetermined degree of deformation is reached.

When the deformation of a crystal is finished, its {111} and {100} poles are either plotted directly in stereographic projection or stored in the drum for later plotting (the plotting of the poles is only accomplished in ROLTEX 1 and ROLTEX 2, not in HORS dOEUVRE).

2.2. The Program ROLTEX 2

The program was written in GIER ALGOL, and the structure of the program is such as to minimize the running time on a GIER computer (minimum number of array calls, drum storing of the plotter data for plotting when all calculations are finished). Later the program has been translated into ILLINOIS ALGOL for use on the IBM 7094. The version listed in this report (Appendix A) is the GIER ALGOL version.

The present description will concentrate on showing how the physical problems of deformation and texture development are translated into computer language. Less attention will be paid to computer-administration problems (such as the problem of plotting the poles in the order that will minimize the computer-time consumption).

2.2.1. Meaning of the General Variables

The underlined variables are input variables, the others are only used inside the computer.

indicator	An integer that governs the program; it can be given values from -2 to 3. The different values correspond to the following operation modes:	
	-1: orientations are given to the computer	no deformation, immediate plotting of the poles
	0: random orientations are generated	
	1: random orientations are generated	

	-2: orientations are given to the computer) deformation before plotting 2 and 3: random orientations are generated) deformation before plotting indicator ≤ 0: plotting in four quadrants indicator ≥ 0: plotting in first quadrant only indicator = 3: apart from plotting, the orientations of the crystals are printed out.
i	The number of crystals.
RANDOM	The maximum number of steps for which the same random stress is allowed to operate.
STOP	The maximum number of steps for each crystal.
randomparameter	An integer that governs the generation of random stresses.
crystalparameter	An integer that governs the generation of random orientations.
counter	An integer used for counting the number of steps.
k, p, q, r, s, t, rr, ss, P, Q, PP, QQ, PQ	Integers used for different purposes throughout the program.
K, Ks12, Ks13, Ks23	Proportionality factors for the shape-restoring stresses; when they are all given the value zero, there is no shape restoration. K is used for tensile and compressive stresses, the others for shear components.
L	A real used for the calculation of the maximum value for the random stresses.
SHOOT	A real used when overshooting is included. When SHOOT < -0.5, the part of the program that takes care of the overshooting is omitted.
R	The final reduction in per cent.
step	The amount of shear in each slip event.
sq, u	Reals used for various purposes.
red	The actual reduction of a crystal during "deformation".
length	The actual length of the crystal considered.
thickness	The actual thickness of the crystal.
transdim	The actual transverse dimension (width) of the crystal.

SA	Used for finding the slip system with the highest resolved shear stress.
fault	Used for calculating the deviation from orthogonality in the crystal lattice after deformation.
A11, A12, A13, A21 ....A33	When starting with a new crystal Apq is the q-co-ordinate of the crystallographic p-axis in the "geometric" co-ordinate system having the transverse, the rolling, and the normal direction as axes (it should be noted that the numbering of the geometric axes in the program is different from the numbering in refs. 2, 3, 6). When dealing with the lattice rotation corresponding to a slip event, Apq is the q-co-ordinate of the "old" geometric p-axis (the p-axis before rotation) in the "new" geometric co-ordinate system. In routine use of the program the Apq's are not input variables, and therefore they are not underlined. Some of them are input variables for indicator < 0.
LL1, LL2, LL3	Reals used for the calculation of the maximum random stress. They are normally supplied from the auxiliary program HORS dOEUVRE.
LLpos	A boolean telling the computer whether the random-stress part of the program can be omitted; it can when LLpos = wrong.
f(1:3, 1:3)	An array in which the shape-restoring stresses are stored; f(p, q) is the restoration-stress component σ_{pq}.
fr(1:3, 1:3)	An array for storing the random stresses (corresponding to f(1:3, 1:3)).
n(1:4, 1:3)	When starting with a new crystal, n(p, q) is the crystallographic q-co-ordinate of the normal to slip plane No. p (the four {111} planes are numbered arbitrarily: (111) is No. 1, $(11\bar{1})$ is No. 2, $(1\bar{1}1)$ is No. 3, $(1\bar{1}\bar{1})$ is No. 4). As the first step in the calculations the crystallographic co-ordinates of the slip-plane normals are converted into geometric co-ordinates, and from then on the n(p, q)'s are geometric co-ordinates.
F(1:4, 1:3)	An array used for calculating the resolved stress across the different slip planes (F(p, q) is the geometric q-component of the stress across slip plane No. p).

d(1:4, 1:3, 1:3)	An array that stores the co-ordinates of the slip directions in the same way as the above n-array stores the slip plane co-ordinates. d(p, q, r) is the r-co-ordinate of the q-th ⟨110⟩ direction lying in slip plane No. p (the three ⟨110⟩ directions in each {111} plane are numbered arbitrarily; each ⟨110⟩ is lying in two different {111} planes, and therefore the numbering p, q, covering 4 x 3 = 12 numbers, will include twice the number of different ⟨110⟩ directions).
rcounter(1:3, 1:3)	rcounter(p, q) indicates the number of slip events for which a certain random stress component σ_{pq} has been operating.
RC(1:3, 1:3)	RC(p, q) indicates the number of slip events for which the random stress component in question will be allowed to operate. RC(p, q) has the maximum value RANDOM.
dr(1:2)	An integer array governing the drum-storing. dr(1) is used for storing the {111} poles, dr(2) for storing the {100} poles.

Apart from the above general variables, different local variables are declared in the program; their meaning will be explained during the detailed program description.

2.2.2. The Procedures Applied

Most procedures in the program are taken from the library of the computer group at Risø.

The procedures head, stop, and CR are part of the standard administration block, ADM 2B.

The plotting procedures (plotline, plotsymbol, plottext, plotaxes, and plotnumber) are described in ref. 7.

The procedures random and randomcryst are slightly modified versions of the library procedure random. The call of random (A, B) or randomcryst (A, B) will provide a pseudo-random number between A and B. After 2796203 calls the same sequence of numbers will reappear. The starting point in the sequence of numbers is governed by the initial values of the variables randomparameter and crystalparameter (governing random and randomcryst respectively).

2.2.3. Co-ordinate Systems and Input

In order to facilitate the understanding of the program description a flow chart is shown in fig. 1. Fig. 1 refers to indicator = 2, i. e. the main operation mode.

The real program begins at "comment now coordinate systems and input;" on page A5. The previous part consists of administration block and declarations.

The computer is given the variables common to all crystals in the problem (a possible input of crystal orientations takes place later), and the plotter paper is provided with co-ordinate systems and text. The co-ordinate systems may be drawn in two different ways depending upon whether four quadrants or only one quadrant are used for plotting (indicator ≤ 0 or indicator > 0 respectively). The local variables cx and cy are used when the text is written.

In order to ensure that the computer has got the correct data, the input data are printed out. The output for a typical problem is shown in Appendix B (the steps/fault output will be described later).

As the final step before going to the individual crystals, dr(1) and dr(2) are given the correct values for governing the drum storing of the {111} and {100} poles respectively. Also t, governing the output, and LLpos are given their proper values.

2.2.4. Generation or Input of Crystal Orientations

At "comment now the crystals one by one;" the program starts working with the individual crystals.

When indicator ≥ 0, random orientations are generated. The geometric x_3-co-ordinate of the crystallographic x_1-axis, A13, is given a random value between 0 and 1 corresponding to the x_1-axes being distributed at random on a hemisphere. When A13 is determined, the position of the crystallographic x_1-axis is fixed to a small circle (that may degenerate into a point or a great circle) as shown in fig. 2. The position on the small circle, characterized by A11 and A12, is determined by giving the angle u (fig. 2) a random value between 0 and $\pi/2$ (there is an extra call of randomcryst to reduce the chance of finding a correlation between A13 and u). The x_1-axis is thus restricted to a solid angle of $\pi/2$ out of the total 4π, but because of the symmetry the 8 parts into which the 4π are divided are all equal. If $A13 \geq 0.99999$, A13 is made equal to 1 and A11 and A12 are made equal to 0.

The crystallographic x_2-axis must lie on a great circle perpendicular to the x_1-axis. The position on this circle is characterized by the angle from the intersection point between the great circle and the rolling plane (the plane containing TD and RD). This angle is given a random value between 0 and π (as was the case above, an extra call of randomcryst has

been inserted). From the angle the geometric co-ordinates of the crystallographic x_2-axis, A21, A22, and A23, are calculated. If the length of the x_2-axis is more than 3×10^{-4} different from unity, the computer prints "-y".

When indicator < 0, the computer reads the values of A11, A12, A13, A22, and A23. From these values A21 is calculated (for this calculation to be possible as formulated in the program, A11 must not be zero, which can always be avoided without introducing any real restrictions to the orientations that can be considered). The x_1-axis and the x_2-axis are finally normed to unity, as the program is only working with vectors of unit length.

The last step, common to all modes of operation, is the calculation of the geometric co-ordinates of the crystallographic x_3-axis, A31, A32, and A33. The calculation is based on the fact that the x_3-axis is the vector product of the x_1- and the x_2-axes. If the length of the x_3-axis is more than 3×10^{-4} different from unity, the computer prints "-z".

2.2.5. Deformation and Lattice Rotation

This part of the program starts at "comment coordinates being generated now slip;". First the different variables including the n(p, q)'s and the d(p, q, r)'s described in 2.2.1, are given the correct value for starting with a new crystal. It should be noted that the shape-restoring stresses and the random stresses are given the initial value zero. The fact that length, thickness, and transdim start with the value 1, means that the crystals considered are initially thought of as being cube-shaped.

The operations down to the label "new slip event:" are only performed once for each crystal. After the label, the operations are repeated (provided the program does not go directly to plotting). The n(p, q)'s and the d(p, q, r)'s are converted into geometric co-ordinates or to "new" geometric co-ordinates, depending upon whether it is the first time the crystal in question passes this part of the program or one of the subsequent times: the Apq's are either the geometric co-ordinates of the crystallographic axes or the "new" geometric co-ordinates of the "old" geometric axes (see 2.2.1). For the co-ordinate conversion the local variables np1, np2, np3, dpq1, dpq2, and dpq3 are used in order to minimize the number of array calls, an array call being a rather lengthy process in the GIER computer. When the conversion has been accomplished, it is decided whether a new slip event shall take place, or the computer go to "plot"; it goes to plot if the numerical value of indicator is less than 1.5, or if the predetermined reduction or number of steps has been reached.

The next step is the calculation of the resolved shear stress for all the slip systems. The shear stress on slip system p, q (see 2.2.1) is calculated from the basic equation

$$Spq = \sum_{i=1}^{3} \sum_{j=1}^{3} n(p, i) \times d(p, q, j) \times \sigma_{ij} . \qquad (1)$$

The meaning of d(p, q, i) and n(p, j) is evident from 2.2.1. σ_{ij} consists of three parts:

$$\sigma_{ij} = \sigma m_{ij} + f(i, j) + fr(i, j), \qquad (2)$$

where the meaning of f(i, j) and fr(i, j) is explained in 2.2.1. σm_{ij} is the main stress component (see 2.1); the main stresses are a tensile stress in the rolling direction (the x_2-axis) and a compressive stress in the normal direction (the x_3-axis), i.e. $\sigma m_{22} = 1$, $\sigma m_{33} = -1$, the rest of the σm's being zero.

An algorithm directly corresponding to (1) would involve a great number of array calls, which should be avoided. Therefore the more complex pattern shown in the program listing has been adopted; Fpq, Fp1, Fp2, Fp3, and Spq serve as local variables. Of the 12 Spq's (resolved shear stresses) the one with the highest numerical value is selected; its number is called P, Q. ø is given the value -1 if the selected Spq is negative, otherwise ø is 1. If the possibility of overshooting is considered (SHOOT $>$ -0.5 or, in order to make it real overshooting, SHOOT $>$ 1), the resolved shear stress for the slip system that was active from the start of the deformation is multiplied by SHOOT as long as it is the only active system; in cases where the same slip system is operating during more than three consecutive slip events at a later stage of deformation, the resolved shear stress for this system will also be multiplied by SHOOT. The selection of slip system is finished at "end p: = 1, 2, 3, 4 end block;" on page A8.

The q/r step in the selection part of the program is also used for controlling whether RC(q, r) is reached for any of the combinations of q and r (see 2.2.1); if this is the case, and if q ≠ r, a new RC(q, r) and a new random stress component fr(q, r) are generated; if q $>$ r, then RC(q, r) = RC(r, q) and fr(q, r) = fr(r, q) as required in order to maintain static equilibrium. The maximum value L for the random stresses is calculated before each slip event from a third-degree polynomial in "red", the coefficients of which are normally supplied from the auxiliary program HORS dOEUVRE.

The slip system being selected, the program is ready for deformation and lattice rotation. First the local variables d1, d2, d3, n1, n2, n3 are declared. They are the co-ordinates of the slip direction and the slip-plane normal. If the numerically highest resolved shear stress is negative, this means that slip must take place in the opposite direction of the original slip direction, and therefor the slip direction is turned 180° ($\phi = -1$).

The calculation of the deformation and the lattice rotation is illustrated in fig. 3. For a start it is assumed that deformation takes place without lattice rotation. The material at the point (0, 0, 0) is supposed not to move, and the first problem is to calculate the displacement of the material in the points (1, 0, 0), (0, 1, 0), and (0, 0, 1). As an example the point (1, 0, 0) (the endpoint of the unity vector $\bar{x}_1$) will be considered. The distance between the slip plane through (0, 0, 0) and the parallel slip plane through (1, 0, 0) is the projection of $\bar{x}_1$ on $\bar{n}$, i. e. $\bar{x}_1 \cdot \bar{n} = n1$. Therefore the material in (1, 0, 0) will be displaced n1 x step in the direction of $\bar{d}$ relatively to (0, 0, 0), step being the shear strain per slip event (2. 2. 1). This means that

material from $\bar{x}_1$ goes to $\bar{x}_1 + (\text{step} \times n1)\bar{d}$,

and by analogy

material from $\bar{x}_2$ goes to $\bar{x}_2 + (\text{step} \times n2)\bar{d}$,

material from $\bar{x}_3$ goes to $\bar{x}_3 + (\text{step} \times n3)\bar{d}$,

still under the preliminary assumption that there is no lattice rotation.

Now there is a lattice rotation fulfilling requirements (a) and (b) in 2. 1. Requirement (a) means that the material from $\bar{x}_2$ can only be displaced along the x_2-direction, i. e. the material preliminarily displaced to $\bar{x}_2 + (\text{step} \times n2)\bar{d}$ must rotate back to the x_2-axis. Requirement (b) means that the material from $\bar{x}_1$ can only be displaced in the x_1x_2-plane, i. e. the material preliminarily displaced to $\bar{x}_1 + (\text{step} \times n1)\bar{d}$ must rotate back to the x_1x_2-plane. The rotation from $\bar{x}_1 + (\text{step} \times n1)\bar{d}$ back to the x_1x_2-plane for instance involves a displacement of

$$-\text{step} \times n1 \times d3 \qquad (3)$$

in the x_3-direction. This together with the corresponding displacements derived from the rotation back to the x_2-axis from $\bar{x}_2 + (\text{step} \times n2)\bar{d}$

$$-\text{step} \times n2 \times d1 \quad \text{along } \bar{x}_1 \qquad (4)$$

and

$$-\text{step} \times n2 \times d3 \quad \text{along } \bar{x}_3 \tag{5}$$

results in the following total displacements:

Material from (1, 0, 0) goes to

$$(1 + \text{step} \times n1 \times d1,\ \text{step} \times n1 \times d2 + \text{step} \times n2 \times d1,\ 0), \tag{6}$$

material from (0, 1, 0) goes to

$$(0,\ 1 + \text{step} \times n2 \times d2,\ 0), \tag{7}$$

material from (0, 0, 1) goes to

$$(\text{step} \times n3 \times d1 + \text{step} \times n1 \times d3,\ \text{step} \times n3 \times d2 + \text{step} \times n2 \times d3,\ 1 + \text{step} \times n3 \times d3). \tag{8}$$

In these expressions second-order corrections have been omitted. Thus they are only exact when step is infinitely small. Of course step must have a reasonable size in practice so that the desired deformation can be achieved in a reasonable number of steps, and a certain inaccuracy must therefore be accepted.

From displacements (6) and (7) the new dimensions of the crystal are calculated:

The new length equals the old length multiplied by (1 + step x n2 x d2),

the new transverse dimension (transdim) equals the old one multiplied by (1 + step x n1 x d1),

the new thickness is calculated from thickness = 1/(length x transdim),

the new reduction is calculated from red = 1-thickness.

The shape-restoring forces are calculated from displacements (6) - (8) (and from transdim derived from the displacements) by changing the sign and multiplying by the relevant K-factor:

f(1, 1) = K x (1-transdim),

f(2, 2) = K x (1-1/transdim),

$$f(1,2) = f(2,1) = \text{old } f(2,1) - Ks12 \times \frac{1}{2} \times (\text{step} \times n1 \times d2 + \text{step} \times n2 \times d1),$$

$$f(1,3) = f(3,1) = \text{old } f(3,1) - Ks13 \times \frac{1}{2} \times (\text{step} \times n1 \times d3 + \text{step} \times n3 \times d1),$$

$$f(2,3) = f(3,2) = \text{old } f(3,2) - Ks23 \times \frac{1}{2} \times (\text{step} \times n2 \times d3 + \text{step} \times n3 \times d2).$$

The shear component f(p, q) must be equal to f(q, p) in order to maintain static equilibrium, whereas one of each pair would come out as zero if directly calculated from the shear displacements. Therefore both components in each pair are made equal to the mean value of the zero and the non-zero value calculated from the displacement, as shown above.

The lattice rotation is introduced into the program by changing the geometric co-ordinate system (see 2.2.1). The "new" co-ordinates Apq of the "old" co-ordinate axes are calculated from rotation displacements (3) - (5) and the displacement along $\bar{x}_1$ and $\bar{x}_2$ (Apq is the new q-co-ordinate of the old p-axis):

A11 = 1 - step x n1 x d1
A12 = step x n2 x d1
A13 = - step x n1 x d3
A21 = - step x n2 x d1
A22 = 1 - step x n2 x d2
A23 = - step x n2 x d3.

The change in sign that was introduced in (3) - (5) is maintained in the above equations for Apq (one might say that there has been a double change in sign, one by going from rotation of the crystal lattice to rotation of the geometric co-ordinate system, another by going from considering the rotation of this co-ordinate system to considering the new co-ordinates of the old co-ordinate axes). In the case of A12 there is no direct rotation component of the x_1-axis from (3) - (5), but only the one brought about by the rotation of the x_2-axis in the x_1-direction, and therefore an extra change in sign takes place, reflected in a positive sign.

The A1q's and the A2q's are normed to unity length of the old $\bar{x}_1$ and $\bar{x}_2$, and the new A3q's are calculated from the fact that the old x_3-axis is the vector product of the old x_1- and the old x_2-axis. Then the Apq's are ready to bring about the change in geometric co-ordinates of the slip planes and slip directions in the next cycle.

At "end block;" the calculations of the change in shape and the lattice rotation have been accomplished, and a new slip event can be performed.

If overshooting is taken into consideration, the number code of the last active slip system must be stored before the jump to a new slip event.

The part of the program in which deformation and lattice rotation take place ends immediately before "comment now pole figure coordinates to drum or to plotter;".

2.2.6. Storing of Plotter Data or Direct Plotting

This part of the program begins at the label "plot:", to which the program jumps when a crystal is not to undergo any further deformation (|indicator| ⟨ 2 or red ⟩ R/100 or counter ≥ STOP). In routine use of the program (indicator ⟩ 0) the plotter data are stored in the drum for later plotting; this makes it possible to minimize the plotter pen movements and hence the plotting time. When indicator ⟩ 0, all the poles are plotted in the first quadrant of the pole figures (making use of the symmetry about the rolling direction and the transverse direction) in order to obtain the maximum density of poles. The direct plotting is only used for special applications of the program such as testing.

The reals cx, cy, xpole, and ypole are used in this part of the program; cx and cy are co-ordinates on the plotter paper, xpole and ypole are co-ordinates in the pole figures (having 1 as the maximum value). The real array COO is used for transferring the plotter paper co-ordinates of a crystal to the drum. When transferring the {111} poles, COO has 8 elements, when transferring the {100} poles, COO has 6 elements.

The slip planes ({111} planes) are treated first. The pole figures are stereographic projections on the $x_1 x_2$-plane. The calculation of the pole-figure co-ordinates is illustrated in fig. 4 (in fig. 4 n(p, 3) is positive). From the two triangles ABO and ACD it is seen that

$$xpole = n(p,1)/(|n(p,3)| + 1).$$

The corresponding expression for ypole is

$$ypole = n(p,2)/(|n(p,3)| + 1).$$

If all the points are to be plotted in the first quadrant (indicator ⟩ 0), the numerical values of xpole and ypole are used for calculating the plotter paper co-ordinates, and the co-ordinates are stored in COO(1:8) when calculated. The pole figure has its centre in (12.5 cm, 37 cm) and a radius of 9.9 cm (this radius is selected to fit the spiral paper of the Philips texture goniometer on which the experimental pole figures are drawn). When the

co-ordinates of all four slip planes have been calculated, the COO array is transferred to the drum (all the {111} co-ordinates are stored in one part of the drum, the {100} co-ordinates being stored in another part).

If indicater ≠ 0, cx and cy are given the value of the pole co-ordinates as above, except that a possible negative sign of xpole or ypole is maintained, and that a possible negative value of n(p, 3) will provoke a change in sign of xpole and ypole. When cx and cy have been given the proper values, the poles are plotted directly.

After "end 111 pole figure;" the corresponding operation for the {100} pole figure is accomplished. The geometric co-ordinates of the {100} poles are not known explicitly, because {100} is not a slip plane and ⟨100⟩ is not a slip direction. They are obtained by combining different slip-plane normals. The real array x(1:3, 1:3) is used for storing the {100} poles, x(p, q) being the qth co-ordinate of {100} pole No. p. The numbering of the slip planes is such that the (100) pole (pole No. 1) for instance may be found as the sum of the poles of slip planes Nos. 1 and 4, meaning that x(1, q) is calculated as n(1, q) + n(4, q),followed by norming to unity.

The procedures for the calculation of the plotter data of the {100} poles and for the storing of these data or for the direct plotting are equivalent to the procedures for the {111} poles, apart from the fact that there are three poles instead of four. Before storing or plotting, the {100} poles are used for calculating the faults that were introduced because second-order terms were neglected in the calculation of the co-ordinates after the individual slip events (cf. p. 14). The faults show up in the angles between the three {100} poles; they will not be exactly 90° (the crystal is not strictly cubic). The numerical cosine of the angle most different from 90° is printed out. At the same time the number of steps performed (counter) is printed. A print-out for indicator = 2 is seen in Appendix B. If indicator = 3, the crystallographic co-ordinates of the rolling-plane normal (the geometric x_3-axis) and the rolling direction (the geometric x_2-axis) are printed out as well; this is the way "ideal orientations" are normally described ({hkl} ⟨uvw⟩). The crystallographic q-co-ordinate of axis No. p is equal to x(q, p).

At "end 100 pole figure end plot; end CRYSTAL;" the storing/plotting operation, and thereby the treatment of the crystal in question, is finished. At "end CENTRAL BLOCK;" the last crystal and, if direct plotting is used, the problem, is finished.

2.2.7. Plotting from Drum

The plotting of the drum data starts at "if indicator)0.5 then begin comment now plot from drum;". This part of the program cannot be fully understood without knowledge of the internal structure of the GIER computer, machine language being mixed up with the ALGOL program. Therefore no attempt will be made to explain how it works in detail. The omission of a detailed description should not influence the reader's basic understanding of what is going on in the program: The calculation of the data stored in the drum has been thoroughly described; this part of the program only performs the plotting of these data in a very fast way. The idea behind the operation is that the poles are taken from the drum 150 at a time, and that these 150 poles are divided into 36 groups, the poles in each group being placed in one of 36 squares covering the area of the pole figure. Then the plotting starts in a square in one corner of the pole figure. When the points in this square have been plotted, the points in the next square are plotted, and so on until the plotting of all 150 points has been accomplished. The method reduces the plotter pen movements between the different points to be plotted.

2.2.8. Running Time

On the GIER computer typical problems as deformation of 100 crystals with 5% shear per slip event (step = 0.05) take approximately 120 min. and 250 min. for 50% and 80% reduction respectively. Of this approximately 20 min. are spent on the plotting operation. The plotter time of 20 min. refers to plotting from drum. If the poles from 100 crystals had to be plotted directly, the time spent on plotting would be about 60 min.

When the IBM 7094 computer at NEUCC Lundtofte was used instead, the above problems would take about 3.25 min. and 7.75 min. respectively (as compated with the 120 min. and 250 min.). In these figures the plotting time is not included, since the plotting was performed by an auxiliary computer so that the main computer did not have to spend time waiting for the mechanical movements of the plotter pen.

2.3. The Program ROLTEX 1

As already mentioned the program ROLTEX 1 has been replaced by ROLTEX 2 for general use. The only reason why ROLTEX 1 is included in this report is that it can be used for a test for which ROLTEX 2 is not suited: For indicator = 3 ROLTEX 1 plots the poles for each step of deformation of crystals for which the initial orientations are given to the computer. In this way the lattice rotations performed by the computer can

be compared with lattice rotations that are easy to foresee theoretically (3.1, 3.2, 3.3). Since the only applications of ROLTEX 1 to be reported refer to indicater = 3, the following short description of the program will only apply to the case where indicator has this value. Before concentrating on indicator = 3 it should be mentioned, however, that one of the main reasons why it was decided to write the new program ROLTEX 2 was the inadequate way in which the "randomly oriented" crystals were introduced in ROLTEX 1.

ROLTEX 1 is listed in Appendix C. It is written in GIER ALGOL, and it has not been translated into ILLINOIS ALGOL. The formal structure is different from that of ROLTEX 2. The main program is rather short, only performing the input and the drawing of the co-ordinate systems. As soon as the computer has got the geometric co-ordinates of the crystallographic x_1- and x_2-axes of a crystal (cf. 2.2.1 and 2.2.4), the procedure SLIP is called. All further manipulations with the crystal are performed in this procedure, and the main program does not get into operation again until a new crystal is introduced.

Since ROLTEX 1 was abandoned for general use at a relatively early stage, the time-saving tricks applied in ROLTEX 2 have not been introduced in ROLTEX 1. For instance the great number of poles to be plotted for each crystal for indicator = 3 strongly suggests the application of drum storing of the plotter data for plotting when all the data have been collected, but, because of the limited use of ROLTEX 1 after the running-in period, it was not found worth-while spending extra time on programming work. The running time for 50% reduction of one crystal in shear steps of 0.05 is more than 30 min. of which the vast majority is plotting time.

Although the formal differences make ROLTEX 1 and ROLTEX 2 look different, there are only the following differences between the two programs as to what they really do to the crystals considered: The shape-restoration proportionality factors Ks12, Ks13, and Ks23 (see 2.2.1 and 2.2.5) are taken together as Ks in ROLTEX 1, meaning that they cannot be varied independently. In the quoted examples the maximum value of the random stress in ROLTEX 1 is constant in each problem (it is equal to twice the input parameter L, the mean value is equal to L); it does not vary with reduction as it does in ROLTEX 2.

Plots for indicator = 3 are shown for instance in fig. 6. The initial orientation is marked with O's, the final orientation is marked with extra big crosses.

2.4. The Program HORS dOEUVRE

HORS dOEUVRE has two functions: It provides the coefficients of the third-degree polynomial used for calculating the maximum value of the random stresses (see 2.2.5), and it can be used for testing the shape restoration (see 3.2).

HORS dOEUVRE is based on ROLTEX 2 (the GIEL ALGOL version), but the final rationalization of ROLTEX 2 has not been transferred to HORS dOEUVRE. As is the case in ROLTEX 1, the shape-restoration proportionality factors Ks12, Ks13, and Ks23 (see 2.2.1 and 2.2.5) have been taken together as one factor, Ks. The main purpose of HORS dOEUVRE is to provide the level of random stresses to be applied in ROLTEX 2. Thus the random-stress level is unknown in HORS dOEUVRE, and therefore random stresses are not included in the program.

The crystals are deformed and rotated as they are in ROLTEX 2, but their final orientations are neither plotted nor printed. Instead average values of the shape-restoration stresses are calculated for different degrees of deformation, and finally the average values are approximated with a third-degree polynomial in red (degree of reduction). The way this is done can be illustrated by the calculation of the average of all the shape-restoring stresses (cf. the listing of the program in Appendix D):

For each deformation step in a given crystal the variable sumf (starting from zero at zero deformation) is increased by the average of the numerical values of the shape-restoration stresses f(1,1), f(2,2), f(1,2), f(1,3), and f(2,3) (see 2.2.1 and 2.2.5). When either K or Ks is very small, the corresponding f-values are nearly zero; the formula with which the average is calculated practically eliminates the influence of such small f's on the result. The integer sumcounter is increased by 1 for each step. At each of a series of degrees of deformation (0,1, 0.2 ..., 0.9) stored in the array RED, sumf/sumcounter is added to the corresponding element in the array SUMF; then sumf and sumcounter are given the value zero so that they can start building up to be included in the next element of SUMF. When one crystal has reached the final degree of reduction, the operations are repeated for the next crystal. In this way the average values of the numerical values of the shape-restoration stresses in the different ranges of deformation (sumf/sumcounter) are summed up for all the crystals in the different elements of SUMF (each element corresponds to a deformation range). When all the crystals have been deformed, the elements of SUMF are divided by the number of crystals (i) to give the average values of the

shape-restoring stresses. These average values are printed and plotted as a function of the reduction. The procedure POLY 1 fits a third-degree polynomial in the reduction (red) to SUMF. The coefficients of this polynomial LL(0:3) are printed, and they are used for calculating the maximum values of the random stresses in ROLTEX 2 (see 2.2.5). The philosophy behind this method of obtaining the random-stress level is treated elsewhere[6]. The third-degree polynomial is plotted together with SUMF-versus-red to show the accuracy of the fit. When the individual crystals were treated, the array RED contained the degrees of deformation for which sumf/sumcounter should be added to SUMF; they were 0.1, 0.2 ... 0.9. Before plotting and fitting, RED is changed to contain the mid-points of the deformation intervals (0.05, 0.15 ..., 0.85). The starting point of SUMF versus red (0, 0) is weighted by the array ZERO so that the polynomial approximation very nearly goes through (0, 0), i.e. LL(0) ~ 0. A plot will be shown in 3.3.

In the same way as SUMF contains the average values of all the shape-restoration stresses, the arrays SUM1, SUM2, and SUM3 sum up the shape-restoration stresses f(1,1) + f(2,2), f(1,2) + f(1,3), and f(2,3) respectively. SUM1, SUM2, and SUM3 are printed, but not plotted and not fitted with a polynomial.

An example of an output sheet for HORS dOEUVRE is given in Appendix E. The input data are printed together with the results (as they are in ROLTEX 1 and ROLTEX 2).

3. THE TESTING OF THE PROGRAMS

For any complicated program like ROLTEX 2 it is essential that the different functions that it is to perform are tested; even if the basic mathematics are correct, there is the risk of programming errors. It is particularly important for ROLTEX 2 because the results that it is primarily intended to produce, viz. calculated textures for a specimen consisting of many grains of initially random orientation, are very difficult or impossible to control directly.

As already mentioned the programs ROLTEX 1 and HORS dOEUVRE are also used in the testing. Needless to say that all precautions are taken to ensure that the test results obtained with the other programs really do test the parts of ROLTEX 2 they are intended to test.

The first tests of the program were of course aimed at checking the trivial functions such as the plotting routines and the basic internal administration of the program, and of course these tests revealed several

errors. However, this is not very interesting, and in the present report we shall restrict ourselves to testing whether the program really is capable of performing a reasonable simulation of the physical processes it is written to simulate. We start at the point where we know that the trivial errors have been corrected.

3.1. Lattice Rotations

The simplest lattice rotation is obtained when the slip direction and the slip-plane normal both lie in the plane determined by the rolling direction and the normal direction (RD and ND). A crystal in such an orientation is shown in stereographic projection in fig. 5. Requirements (a) and (b) in 2.1 will in this crystal lead to a simple rotation with the transverse direction (TD) as the rotation axis. In fig. 6 the rotations of the {111} and the {100} poles (calculated with ROLTEX 1) are shown with a Wulff net superimposed. The circles show the initial poles, the big crosses show the poles in the final orientation, and the small crosses show the intermediate steps. The initial lattice rotation is seen to be a simple rotation about TD. The rotation in fig. 6 is produced by the basic stress system only, i. e. without additional and random stresses. The reduction corresponding to the rotation path in fig. 6 is 45%, and the shear steps are 0.05.

In addition to the qualitative check of the type of rotation, fig. 6 can also be used for quantitative checks. At a certain point the rotation changes because another slip system reaches a resolved shear stress that is higher than that on the original system. In the last orientation of the simple rotation the resolved shear stress on the original slip system is found to be 0.799, whereas the most heavily loaded of the other slip systems has a resolved shear stress of 0.814. This means that we have passed the point where a new slip system should be activated, and the computer reacts accordingly: The new system takes over in the next step as revealed by the change in lattice rotation. In the orientation after the last but one step in the simple rotation both resolved shear stresses are found to be 0.809 (the calculations are performed on the basis of the plotted orientations, meaning that they are not very accurate).

The initial simple rotation path in fig. 6 consists of 23 steps (it may not be clear from fig. 6 as reproduced, but the steps can be counted on the original computer plot). The number of steps can be related to the rotation angle. Fig. 7 shows a section of the gliding crystal after a shear increment, $d\dot{s}$, before the corresponding lattice rotation has taken place. The lattice rotation will bring AB back to the rolling direction, RD, meaning that the

angle v will be reduced by the incremental angle BAC, i.e. $\angle$ BAC is -dv. From simple trigonometry we get the following results:

From triangle ACD:

$$AC = \frac{1}{\sin v} .$$

From triangle ABC:

$$\frac{BC}{\sin(-dv)} = \frac{AB}{\sin(180^\circ - v)}$$

$$\Rightarrow \frac{BC}{-dv} \simeq \frac{AC}{\sin(180^\circ - v)}$$

$$\Rightarrow \frac{ds}{-dv} \simeq \frac{AC}{\sin(180^\circ - v)} = \frac{1}{\sin^2 v}$$

$$\Rightarrow ds \simeq \frac{-dv}{\sin^2 v} .$$

The total shear, s, should therefore be related to the initial and the final value of v for the simple rotation with the expression

$$s = -\int_{v_i}^{v_f} \frac{dv}{\sin^2 v} = \cot v_f - \cot v_i .$$

s is 23 steps of 0.05 which is equal to 1.150. From fig. 6 v_f and v_i can be found to be approximately 27 and 51° (from the {111} pole moving along RD), i.e. $\cot v_f - \cot v_i$ is equal to 1.153. This means that within the accuracy with which we can read the angles the computer reproduces the correct relation between shear and rotation angle.

The elongation in the rolling direction taking place during the simple lattice rotation can be illustrated in fig. 7. If s is substituted for ds, the original AC of length $\frac{1}{\sin v_i}$ is elongated to AB of length $\frac{1}{\sin v_f}$, meaning that an original length 1 will be elongated to $\frac{\sin v_i}{\sin v_f}$. Since the simple shear considered has no component in the transverse direction such an elongation will correspond to a reduction of an original length 1 in the normal direction to a length of $\frac{\sin v_f}{\sin v_i}$. The reduction during the simple lattice rotation should therefore be

$$1 - \frac{\sin v_f}{\sin v_i} = 1 - \frac{\sin 27^\circ}{\sin 51^\circ} = 0.416 = 41.6\%.$$

The total reduction in fig. 5 is 45%, and there are only few steps after the change in rotation path, which agrees with the above value of 41.6%. In order to get a more accurate check we have given the computer another problem with conditions identical to those in fig. 5 apart from a change in reduction from 45% to 41.6%. The computer performs the 23 steps of the simple rotation plus one extra step, which reflects a perfect agreement within the inaccuracy with which we read v_f and v_i. Thus, the computer also reproduces the correct relation between reduction and rotation angle.

In a special version of ROLTEX 1, ROLTEX 1A, the basic stress system has been changed; it consists of a tensile stress in the transverse direction and a compressive stress in the normal direction and no stress in the rolling direction. Such a stress will obviously not produce a rolling deformation in the rolling direction, but it is useful for checking the lattice rotation. In a crystal with slip direction and slip-plane normal lying in the plane determined by TD and ND this special basic stress system will together with requirements (a) and (b) produce a simple lattice rotation with RD as the rotation axis. The desired orientation can be obtained by rotating the crystal in fig. 5 90° about ND, and fig. 8 shows the rotation of the {111} poles of this crystal brought about by the special stresses (from now on the rotation of the {100} poles will not be shown). The conditions in fig. 8 are the same as those in fig. 6 apart from the initial 90° rotation of the crystal, from the change in basic stresses, and from a change to a total reduction of 50% instead of 45%. The initial lattice rotation is seen to be a simple rotation about RD as it should be, and the duration of this rotation path is seen to be the same as that in fig. 6, as should also be expected. The "non-simple" rotation path is longer in fig. 8 than in fig. 6, because the reduction is 50% instead of 45%.

The rotation path of any pole can be composed of a path corresponding to a rotation about TD and a path corresponding to a rotation about RD, and we have now seen examples in which these two rotations work correctly, i.e. there are good reasons to have confidence in the lattice rotations produced by the programs.

It is easily seen that certain symmetrical orientations are stable because the lattice rotations produced by symmetrically oriented slip systems cancel each other. We shall consider the orientation {110} ⟨211⟩. Fig. 9

shows the stereographic projection for this orientation with the active slip systems indicated, and it shows the {111} poles produced by ROLTEX 1 plotting the poles for each step during 25% reduction with the basic stress system only. An oscillation between two orientations very close to each other is seen corresponding to the two active slip systems. In this connection only the small crosses should be considered because there is a slight inaccuracy in the plotting of the big crosses. The stable orientation is actually the main orientation of the brass-type texture.

Fig. 10a shows another symmetrical orientation, {110} ⟨100⟩. There are four symmetrically oriented primary slip systems and if they operate simultaneously the orientation is stable, but it is an unstable equilibrium. As soon as one of the systems is more active than the others, the orientation will start changing, and in the computer this happens already in the first step. Let us assume that the slip system marked with full circles operates in the first step. This will produce a lattice rotation during which the slip-plane normal will approach the normal direction and the slip direction will approach the rolling direction. Such a lattice rotation can be described by a rotation about TD combined with a rotation about ND, as is indicated with arrows in fig. 10a. The resulting orientation is not symmetrical and there will be one single slip system carrying the highest resolved shear stress, the one marked with half circles. Slip on this system inverts the rotation about TD, but it continues the rotation about ND, meaning that the resulting oscillation between the two marked slip systems will produce a simple rotation about ND. Fig. 10b shows the rotations paths of the {111} poles of the crystal in fig. 10a deformed to 25% reduction in shear steps of 0.05 in ROLTEX 1. The lattice rotation is a simple rotation about ND as foreseen, i.e. the computer is capable of performing simple rotations about the three main axes correctly.

Thus in all four cases considered the lattice rotation (or the lack of lattice rotation) produced by the computer is in qualitative and quantitative agreement with the known correct lattice rotation, which is convincing evidence that the computer will also produce correct lattice rotations in the normal, less simple cases.

3.2. Additional Stresses

The auxiliary program HORS dOEUVRE calculates the mean values for a number of crystals of the numerical values of the individual additional stress components and the mean value of the components taken together as a function of reduction. The main purpose of this is to supply data for the

calculation of the random stresses in ROLTEX 2, but it is also useful for checking that the shape restoration process works.

Statistically complete shape restoration should be obtained when all the K-parameters (K and Ks in HORS dOEUVRE) are given appropriate values. Complete restoration means that the individual grain follows the macroscopic deformation, i. e. a cube is deformed to the shape of a right parallelepiped with the transverse dimension equal to the original cube edge. The additional stresses build up as a reaction to deviations from this shape with a size equal to the deviation multiplied by the relevant K-parameter. If the mean additional stresses provided by HORS dOEUVRE are divided by the corresponding K, the result is the deviation from the wanted shape. Table I gives the mean values of all the additional stresses taken numerically in 20 grains of a certain random orientation distribution during deformation to 80% reduction in shear steps of 0.05, and it gives these mean values divided by K (= Ks). Two cases are considered: K = Ks = 20 which is relevant when shape restoration is wanted (see below), and K = Ks = 0.001 which does not lead to any shape restoration because the additional stresses are too small as compared with the basic stresses (equal to unity). For K = Ks = 20 the mean additional stress divided by K is seen to remain small - of the order of 0.01 - meaning that the average deviation from the ideal shape is of the order of 1%, i. e. the additional stresses produce the intended shape restoration. The total output for K = Ks = 20 is shown in Appendix E. The first vertical column gives the average of all the average additional stress components (quoted in table I). The other columns give the average of the individual additional stress components as described in 2.4. For K = Ks = 0.001 the mean additional stress divided by K increases with deformation as a natural consequence of the fact that there are no additional stresses of such a size that they can force the individual grains to follow the macroscopic deformation. It is worth noting on the other hand that at 50% reduction the mean deviation from the macroscopic shape is no more than 0.113 or 11% (the average of the values for 45% and 55%).

In order to make the additional stresses work properly one must adjust the value of the proportionality factors (the K-parameters) between shape deviation and the resulting additional stress to the magnitude of the shear steps. The magnitude of the steps is an upper limit for the deviation in shape that can result from one step. Step magnitude multiplied by K-parameter is therefore a measure of the largest additional stress that can develop as a result of one step. The average additional stress developed per step is about one order of magnitude smaller. A typical shear step will

be 0.05; if that is used together with K-parameters of 200 for instance, the maximum additional stress will be 10 and the typical additional stress will be 1, i.e. the additional stresses will be bigger than or of the same order as the basic stresses. This will not allow the deformation to take place in a reasonable way: The selection of slip systems will mainly be governed by the additional stresses built up in the preceding shear step so that the deformation will be a series of overreactions and hence different from the rolling deformation that it is intended to simulate. On the other hand the K-parameters must of course have a certain size if they are to produce additional stresses that can influence the slip process and thereby lead to shape restoration, as exemplified in table I. Thus, the additional stresses will fulfil their purpose only for a certain range of K-parameters. This is illustrated in fig. 11 showing the {111} poles of the same group of randomly oriented crystals after 60% deformation in shear steps of 0.05 with K-values varying from 0 to 200. The simulation is performed by ROLTEX 2 with indicator equal to 3 so that the final orientations (expressed as (hkl) [uvw]) are printed; they are given in table II. It is seen that the results for K = 10 and K = 25 are almost identical for all crystals but No. 9, and those for K = 50 are similar to those for 10 and 25 apart from crystal No. 9. The results for K = 0 are somewhat different, but they still correspond to a sharp texture (the brass type[1-3,6]). As mentioned in 2.2.6, the deformation of a crystal is stopped when the wanted deformation is reached or, in order to avoid waste of computer time, when a certain number of steps, STOP, have been performed. In this series of computer experiments the maximum number of steps allowed is 100, which was never reached in the above cases. In the case of K = 200 it is reached for all 10 crystals, meaning that none of them has reached 60% deformation in 100 steps (or, to be exact, in 99 steps). The reason for this is the overreactions described above. The overreactions and the corresponding deviation from the systematic rolling deformation are also reflected in the lack of a sharp texture for K = 200. The conclusion is that for shear steps of 0.05 the additional stresses function as they are intended to in a K-parameter range from 10 to 50, and the results do not depend critically on the values of the K-parameter as long as they are within this range. 0.05 is the biggest shear step applied; step sizes smaller than 0.05 will allow the application of K-values bigger than 50. We have standardized a value of 20 for the K-parameters that are intended to produce additional stresses.

Finally the correct functioning of the additional stresses will be illustrated by two simple examples. The orientation in fig. 5 was shown

to perform an initial simple rotation about TD when no additional stresses were applied (fig. 6). During the initial stage of deformation both slip direction and slip-plane normal lie in the plane determined by RD and ND, which means that shape restoration is automatical for all strain components but one; the only deviation from the ideal shape is that the angle between the edges that were originally parallel to RD and ND respectively, is not maintained right. However, when the slip direction forms an angle of 45° with RD, even this deviation from the ideal change in shape disappears. In fig. 5 the angle is about 50°, and the simple rotation will for a start make it approach 45°. This means that the initial lattice rotation from the orientation in fig. 5 is a simple rotation about TD also when additional stresses are applied as shown in fig. 12, but the simple rotation path is shorter than that in fig. 6: When the angle passes 45° and gets smaller and smaller, additional stresses will build up, and at a certain stage, earlier than in fig. 6, the lattice rotation will change. Fig. 12 refers to 50% reduction in shear steps of 0.05 with K = Ks = 25.

In 3.1 we saw that the {110} ⟨100⟩ orientation (fig. 10) was not stable when deformed with the application of the basic stresses only. It would be if all four symmetrically oriented slip systems operated simultaneously, but with the basic stress system this symmetrical slip pattern is replaced by an oscillation between two slip systems with a resulting lattice rotation as soon as one of the four systems is selected for the first shear step. However, the resulting change in shape does not agree with the macroscopic deformation, and therefore the double slip does not work when additional stresses are applied. Fig. 13 shows the {111} poles of a crystal of {110} ⟨100⟩ orientation deformed to 25% reduction with additional stresses in ROLTEX 1 plotting the poles for each shear step of 0.05. The orientation is stable because the additional stresses force all four symmetrical slip systems to be active (for each of the two slip planes the resultant of the two slip directions lies in the plane determined by RD and ND, i.e. there are two opposite rotations about TD (cf. fig. 6)).

3.3. Random Stresses

As an illustration of the effect of the application of random stresses we shall first consider a case with a very high level of random stresses. The initial orientation is that of fig. 5; this crystal is deformed in ROLTEX1 with the mean value of the numerical value of the random stress components equal to 1, which is the same as the two components of the basic stress (there are no additional stresses). The resulting lattice rotation path is

illustrated in fig. 14. It is seen to be quite different from that in fig. 6 produced without random stresses. The random stresses in fig. 14 are so high that the lattice rotation is mainly governed by them. Such a situation is not realistic for simulation of the physical events, but fig. 14 demonstrates that the random stresses work. It is worth noting that the predetermined reduction of 50% is finally reached, meaning that the permanent action of the basic stresses does produce something like a rolling deformation even when they are jammed by high random stresses, because the average of each random stress component is near zero over a sufficiently large number of steps. Of course the number of steps necessary to produce 50% reduction (more than 100) is much bigger than that necessary to produce 45% without random stresses in fig. 6 (28). This difference is partly due to the fact that the shear steps are only 0.025 in fig. 13 compared with 0.05 in fig. 6, meaning that the correct numbers to be compared are 2 x 28 = 56 and more than 100, but this is still a large difference.

Fig. 15 corresponds to a level of random stresses closer to that used in practice. A crystal of the stable orientation {110} ⟨211⟩ from fig. 9a is given 75 shear steps of 0.025 in ROLTEX 1 with an average random stress of 0.5 and no additional stresses. Now the basic stresses are dominant as revealed by the approximate stability of the orientation, but because of the random stresses there is a substantially larger spread around the stable orientation than there was in fig. 9b. This indicates that reasonable levels of random stresses do not cause any drastic change in the orientations of the deformed crystals, but only an increased orientation spread. The examples shown in chapter 3 in ref. 2 confirm this conclusion.

As already mentioned, the level of random stresses used in practice is provided by HORS dOEUVRE as the mean numerical value of the additional stresses. A typical set of average additional stresses was given in table I for K = Ks = 20. They are seen to be about 0.15 - 0.20. HORS dOEUVRE plots the average additional stress versus reduction as shown in fig. 16 and approximates the average additional stress with a third-degree polynomial in red (also shown in fig. 16). The third-degree approximation is forced to pass through (0, 0), which requires a sharper bending of the curve than the third-degree polynomial can provide, and therefore the polynomial is not a very good approximation. However, the exact level of random stresses is not very critical so that the third-degree polynomial approximation will give sufficient accuracy. HORS dOEUVRE prints out the coefficients of the polynomial, LL(0) - LL(3), as shown in the HORS dOEUVRE output in Appendix E. Three of them are used for the determi-

nation of the random stress level in ROLTEX 2 (LL(0) is very close to zero). The LL's produced by HORS dOEUVRE correspond to the mean value of the additional stresses, and this mean additional stress should define the mean random stress in ROLTEX 2. The polynomial L in ROLTEX 2 with the coefficients LL1, LL2, and LL3 actually defines the maximum value of the random stresses equal to twice the mean random stress (see 2.2.1), and therefore the LL's from the HORS dOEUVRE output should be multiplied by two before they are used as LL's in the ROLTEX 2 input. The plot in fig. 16 shows that the mean additional stress is not too far from being constant in the whole deformation range, meaning that one might use a constant value of the maximum random stress in ROLTEX 2. This is done for certain applications in a special version of the program.

Apart from the maximum value of the random stress components another parameter, RANDOM (see 2.2.2), must also be defined for the governing of the random stresses. For shear steps of 0.05 RANDOM is normally given a value of 10, meaning that the maximum number of steps for which the same random stress component can operate is 10. This corresponds to a mean "lifetime" of 5 steps, i.e. in average each random component will change 20 times in each crystal during deformation to 80% reduction (the average number of steps is about 100, cf. Appendix B).

3.4. The Random Orientation Distribution

In the vast majority of applications ROLTEX 2 is intended to simulate the texture development in an initially texture-free material. As already mentioned, the computer generates pseudo-random orientation distributions for indicator = 0. If these distributions really correspond to random distributions, the "specimens" in the computer are approximately texture-free.

Fig. 17 shows the distribution of {111} and {100} poles of 100 computer-generated crystals (indicator = 1, i.e. there is no deformation). The distribution is not, and should not be, quite even for a random distribution of only 400 and 300 poles. The $\pi/2$ solid angle of the quarter pole figures is divided into 16 areas each corresponding to the same solid angle and each characterized by one of the letters A, B, C, or D combined with one of the figures 1, 2, 3, or 4 as illustrated in fig. 17. The area next to the rolling direction is for instance D4. The number of poles in each of the 16 areas are listed in table III for three different distributions of which the distribution from fig. 17 is No. 1. Table III also gives the number of poles for the areas taken together in groups of four, each group corresponding to a letter or to a figure, and it gives the sums for the three orientation distributions. The

theoretical standard deviations are calculated as the square root of the average number of poles in the type of area considered - based on the preliminary assumption that the distribution function for the number of poles per area corresponds to that of a typical random phenomenon as the number of radioactive decays per unit of time. If the distributions are random in this way, 68% of the numbers should be within the quoted standard deviations (and 96% within twice the standard deviation). Table III shows no tendency for any of the areas to deviate from the average in a systematic way. Out of the 192 numbers in the table 32 are outside the range defined by mean value $\pm$ standard deviation and 2 are outside the range mean value $\pm$ twice the standard deviation. The expected numbers are 61 and 8 respectively, i.e. the actual distribution of the number of poles per area is narrower than the "radioactive distribution". This can be understood from quantitative calculations of the probabilities, but the important thing is that the distribution is relatively narrow, and therefore it is not necessary to go into these calculations.

Thus, the {111} and {100} poles are distributed in space with no systematic deviation from an even distribution. It is therefore justified to conclude that the orientation distributions generated in the computer are satisfactory simulations of texture-free materials. In this connection it should be remembered that even in real materials with their very large number of crystals the pole distributions are normally not quite even.

The facts that the orientation distributions generated are not quite even and that they are uneven in different ways for different values of crystal-parameter have a certain effect on the calculated textures. Fig. 18 shows the calculated {111} pole figures for two different orientation distributions subjected to identical deformation conditions (50% reduction). There is a certain difference between the two pole distributions after deformation, but they obviously correspond to the same type of texture (the copper type as illustrated by the experimental copper pole figure superimposed). In ref. 5 the distribution of discrete orientations produced by ROLTEX 2 was transformed to a smooth three-dimensional orientation distribution, and it was found that two different orientation distributions deformed to 95% reduction under identical conditions produced identical three-dimensional orientation distributions. This shows that the difference between different orientation distributions only exists when the calculated texture is considered as being composed of orientations of individual grains. As soon as the smearing operation is applied (which does not remove systematic differences in texture), the difference disappears.

3.5. Miscellaneous

As mentioned in 2.2.5 and 2.2.6 the crystals are not strictly cubic after deformation, because second-order terms are neglected in the calculation of the lattice rotations and the magnitude of the shear steps must have a finite size for practical reasons.

For each of the 100 crystals deformed to 80% reduction in shear steps of 0.05 the output in Appendix B gives the cosine of the angle between two crystallographic axes that is most different from 90°. For the majority of the crystals the cosine is smaller than 0.01 corresponding to deviations of approximately half a degree or less. The mean value of the ten largest deviations is 0.8°. The situation reflected in Appendix B is considered to be just acceptable. Once the deviation has reached a certain level it seems to accelerate, and therefore shear steps of 0.05 are not used for reductions higher than 80%. For 95% reduction the steps are reduced to 0.03. When the shear per step is decreased and when random stresses are applied, the value of RANDOM is normally increased so that the mean lifetime of each random stress component still corresponds to the same amount of shear (see 3.3).

The possibility of including overshooting was described in 2.2.5. It turns out that the application of overshooting does not lead to improved calculated pole figures, and therefore it is not very important. However, the correct functioning of the overshooting procedure may contribute to demonstrating that the program as a whole works properly. The orientation in fig. 9a was found to be stable because of the operation of the two symmetrically oriented slip systems. If overshooting is added, this stability disappears. One of the two systems is selected for the first shear step, and in the next step this system will have its resolved shear stress multiplied by SHOOT and therefore continue to operate with its resolved shear stress becoming increasingly smaller than that on the other system. Eventually the crystal will reach an orientation where the resolved shear stress on the first slip system multiplied by SHOOT will be smaller than that on one of the other systems. If SHOOT is not too big, this system will be the other of the two systems marked in fig. 9a, and the result will be a lattice rotation towards the vicinity of the original orientation. When this is reached, the operative slip system again has the overshooting advantage, and the lattice rotation will go on so that the symmetrical orientation is passed. This orientation oscillation will continue indefinitely. Fig. 19 shows the orientation of the {111} poles after 10, 20, 30, 40, 50, 60, and 70 shear steps of 0.05 performed in ROLTEX 2 with SHOOT = 1.33 (no additional or

random stresses). The corresponding number of steps is indicated for each pole. The crystal is seen to perform the predicted oscillation - with an amplitude of about 10^{o}.

4. CONCLUDING REMARKS

Chapter 2 contains a thorough description of the functioning of the program ROLTEX 2 and the relevant parts of the auxiliary programs ROLTEX 1 and HORS dOEUVRE. In chapter 3 it is shown that the programs work as intended. Thus, the present report supplements refs. 1-6, giving the physical significance of the results obtained with ROLTEX 2: It gives a detailed description of the mathematical means by which the physically significant results are obtained, and thereby it can give the reader an increased possibility of understanding exactly what are the basic assumptions.

ACKNOWLEDGEMENTS

The author wants to acknowledge the assistance of the computer group at Risø in the programming work, particularly in the attempts to rationalize ROLTEX 2. Thanks are also due to P. Nielsen who did the delicate work with the reproduction of the computer plots, and to V. Vegenfeldt for making the drawings.

REFERENCES

1) T. Leffers, Phys. Stat. Sol. 25 (1968) 337-344.

2) T. Leffers, Risø Report No. 184 (1968) 33 pp.

3) T. Leffers, in: J. Grewen and G. Wassermann (editors), Textures in Research and Practice (Springer, 1969) 120-129.

4) T. Leffers, Z. Metallkde. 60 (1969) 535.

5) H.J. Bunge and T. Leffers, Scripta Met. 5 (1971) 143-149.

6) T. Leffers, Risø Report to be published.

7) L. Hansson, Risø-M-551 (1967) 15 pp.

Table I

Mean additional stress and mean deviation from macroscopic change in shape for K = Ks = 20 and for K = Ks = 0.001

Reduction (%)	K = Ks	Mean additional stress	(Mean additional stress)/K
5		0.149	0.0075
15		0.178	0.0089
25		0.197	0.0099
35	20	0.213	0.0107
45		0.213	0.0107
55		0.223	0.0112
65		0.198	0.0099
75		0.198	0.0099
5		1.99×10^{-5}	0.0199
15		4.11×10^{-5}	0.0411
25		6.05×10^{-5}	0.0605
35	0.001	8.02×10^{-5}	0.0802
45		1.01×10^{-4}	0.101
55		1.26×10^{-4}	0.126
65		1.55×10^{-4}	0.155
75		1.89×10^{-4}	0.189

Table II

Number of steps performed (maximum allowed number is 100) and final orientations expressed as (hkl) uvw for the same 10 crystals of initially random orientation deformed in shear steps of 0.05 with the application of different values of the K-parameters

K=Ks12= Ks13=Ks23	Number of steps	h	k	l	u	v	w
0	44	0.762	-0.069	0.646	0.393	0.838	-0.374
	40	0.618	-0.030	-0.786	0.024	-0.998	0.058
	49	0.180	0.808	-0.561	0.797	-0.453	-0.398
	58	0.847	0.483	-0.227	0.413	-0.322	0.850
	48	0.528	0.181	0.830	0.268	0.889	-0.366
	45	0.783	-0.116	-0.612	-0.336	-0.904	-0.256
	49	0.126	0.785	-0.607	0.802	-0.441	-0.402
	45	0.022	0.699	0.716	0.648	0.535	-0.543
	44	0.817	-0.181	-0.548	-0.409	-0.851	-0.328
	43	0.752	0.659	0.061	0.478	-0.480	-0.735
10	52	0.800	-0.105	0.593	0.389	0.839	-0.377
	43	0.617	-0.005	-0.787	0.023	-0.999	0.025
	61	0.268	0.884	-0.384	0.700	-0.452	-0.552
	61	0.896	0.365	-0.256	0.429	-0.553	0.713
	51	0.263	0.122	0.957	0.138	0.976	-0.162
	43	0.788	-0.032	-0.616	-0.219	-0.946	-0.230
	63	0.240	0.875	-0.420	0.727	-0.449	-0.519
	68	0.321	0.354	0.878	0.643	0.600	-0.477
	48	0.862	-0.163	-0.481	-0.402	-0.798	-0.448
	63	0.588	0.795	-0.158	0.478	-0.498	-0.723
25	54	0.763	-0.063	0.646	0.388	0.839	-0.377
	44	0.605	0.010	-0.796	-0.007	-1.000	-0.018
	62	0.255	0.890	-0.379	0.700	-0.440	-0.563
	60	0.893	0.366	-0.262	0.432	-0.534	0.725
	48	0.340	0.108	0.934	0.126	0.979	-0.159
	43	0.788	-0.023	-0.616	-0.174	-0.965	-0.187
	63	0.217	0.874	-0.436	0.747	-0.436	-0.501
	68	0.301	0.361	0.883	0.640	0.609	-0.468
	50	0.946	0.291	0.150	0.318	-0.738	-0.597
	65	0.557	0.817	-0.155	0.499	-0.478	-0.723
50	57	0.704	0.029	0.711	0.362	0.843	-0.393
	45	0.595	0.004	-0.803	0.008	-1.000	0.001
	63	0.262	0.885	-0.386	0.687	-0.451	-0.569
	61	0.913	0.301	-0.275	0.399	-0.518	0.757
	50	0.542	0.059	0.839	0.253	0.939	-0.230
	45	0.806	-0.023	-0.591	-0.096	-0.991	-0.092
	63	0.219	0.874	-0.435	0.733	-0.441	-0.517
	69	0.297	0.358	0.885	0.652	0.601	-0.462
	56	0.904	0.357	0.238	0.420	-0.639	-0.645
	71	0.635	0.769	-0.090	0.480	-0.484	-0.731

(to be continued)

Table II (continued)

K=Ks12= Ks13=Ks23	Number of steps	h	k	l	u	v	w
200	100	0.803	0.324	0.501	0.076	0.778	-0.624
	100	0.544	0.006	-0.840	0.012	-1.000	0.001
	100	0.024	0.968	-0.248	0.616	-0.212	-0.760
	100	0.989	-0.030	-0.150	0.128	-0.374	0.926
	100	0.078	-0.060	0.995	0.231	0.968	0.041
	100	0.921	0.038	-0.388	-0.039	-0.985	-0.187
	100	0.104	0.954	-0.279	0.785	-0.253	-0.569
	100	0.465	0.634	0.618	0.410	0.465	-0.785
	100	0.939	0.242	-0.244	0.064	-0.819	-0.570
	100	0.724	0.503	-0.471	0.184	-0.798	-0.573

Table III

Distribution of poles in three different orientation distributions

	Distribution 1		Distribution 2		Distribution 3		1 + 2 + 3	
	{111}	{100}	{111}	{100}	{111}	{100}	{111}	{100}
Total number counted[x)]	393	298	392	297	391	291	1176	886
Average number per small area	24.6±5.0	18.6±4.3	24.5±4.9	18.6±4.3	24.4±4.9	18.2±4.3	73.5±8.6	55.4±7.4
Average number per bigger area	98.2±9.9	74.5±8.6	98.0±9.9	74.2±8.6	97.7±9.9	72.7±8.5	294.0±17.1	221.5±14.9
A1	23	17	21	22	21	14	65	53
A2	29	16	26	23	22	21	77	60
A3	25	15	25	14	27	19	77	48
A4	23	21	24	20	25	17	72	58
B1	27	24	25	14	24	13	76	51
B2	23	20	27	26	28	21	78	67
B3	26	22	26	10	23	17	75	49
B4	20	20	28	18	25	17	73	55
C1	22	17	23	20	33	15	78	52
C2	21	19	29	21	27	19	77	59
C3	20	17	28	11	19	24	67	52
C4	23	17	19	23	26	15	68	55
D1	30	19	24	21	23	19	77	59
D2	22	19	23	13	15	24	60	56
D3	32	18	21	21	17	26	70	65
D4	27	17	23	20	36	10	86	47
A	100	69	96	79	95	71	291	219
B	96	86	106	68	100	68	302	222
C	86	70	99	75	105	73	290	218
D	111	73	91	75	91	79	293	227
1	102	77	93	77	101	61	296	215
2	95	74	105	83	92	85	292	242
3	103	72	100	56	86	86	289	214
4	93	75	94	81	112	59	299	215

x) The numbers counted are slightly below the correct numbers. For instance the first one should have been 400 instead of 393. The reason is that two poles may be so close to each other that they are only counted for one.

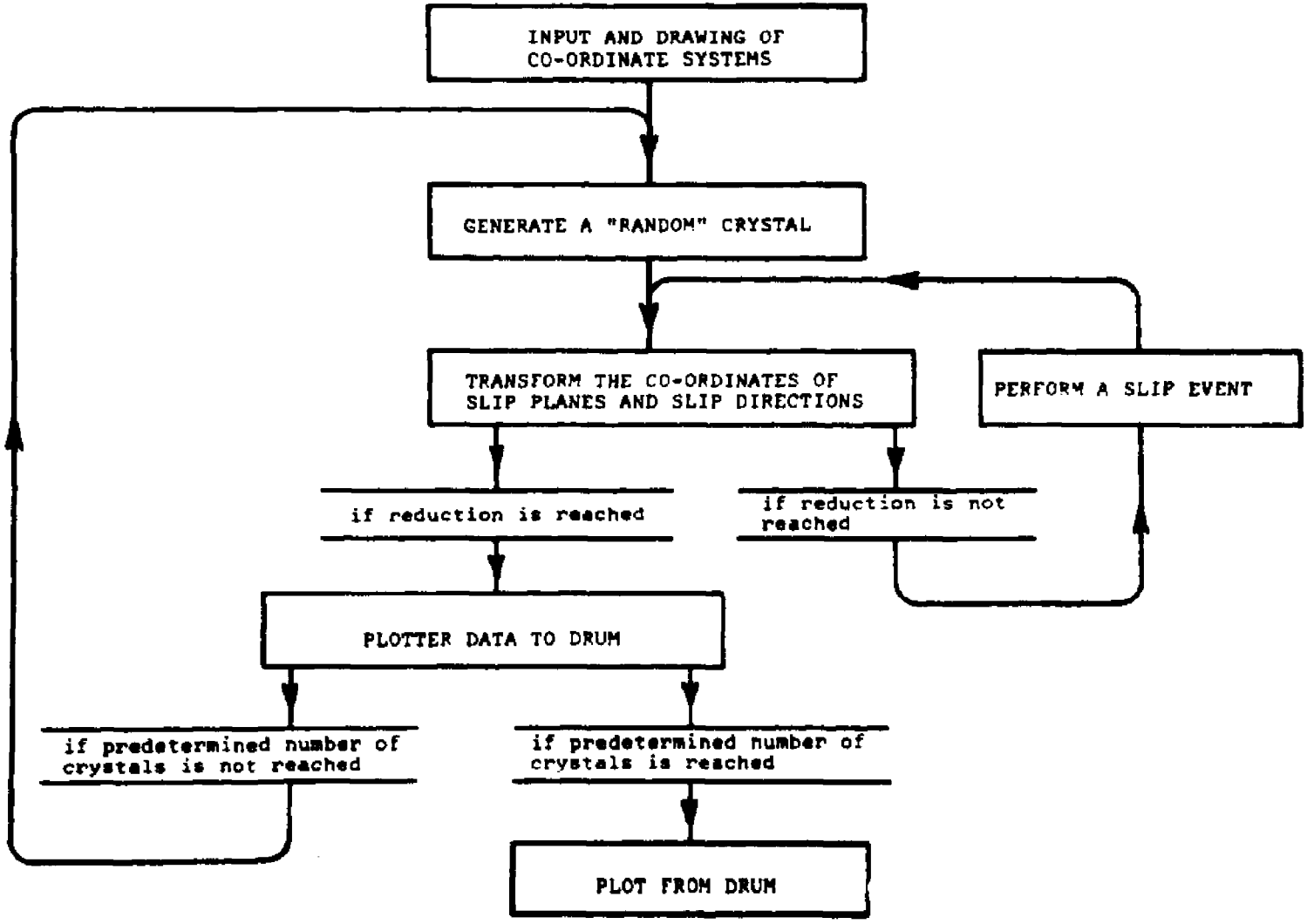

Fig. 1. Flow chart for ROLTEX 2 with indicator = 2.

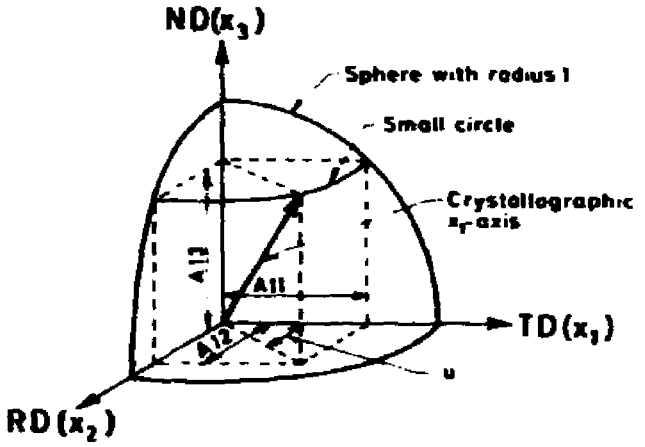

Fig. 2. The generation of a random crystallographic x_1-axis.

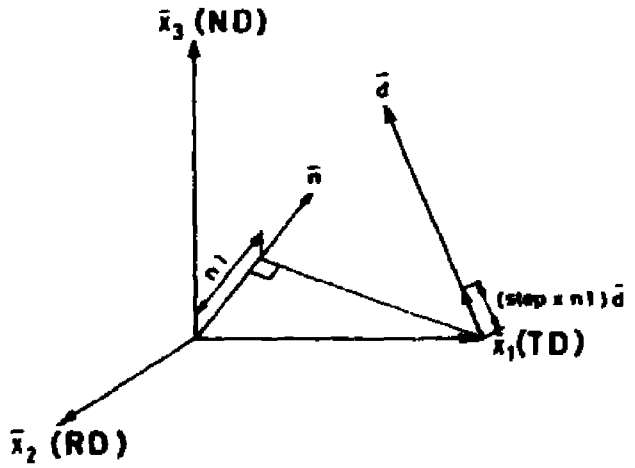

Fig. 3. A geometric co-ordinate system in a deforming crystal. The material in the point (0, 0, 0) is assumed not to move. The figure refers to the situation before lattice rotation.

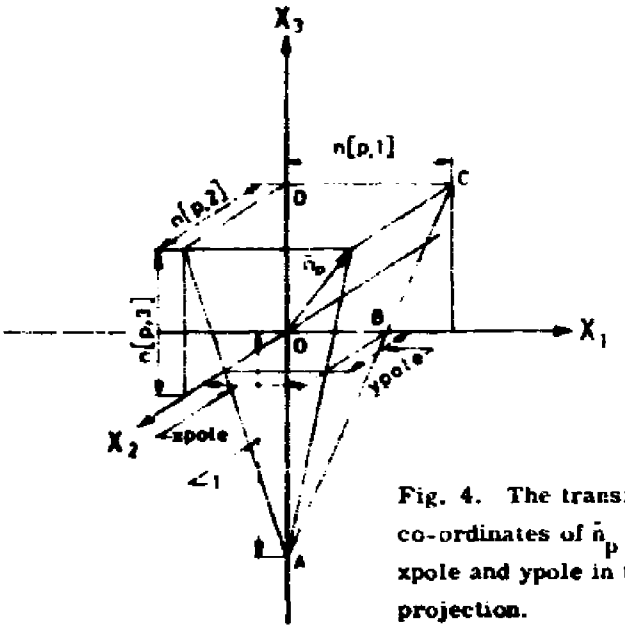

Fig. 4. The transformation of geometric co-ordinates of $\bar{n}_p$ to the co-ordinates xpole and ypole in the stereographic projection.

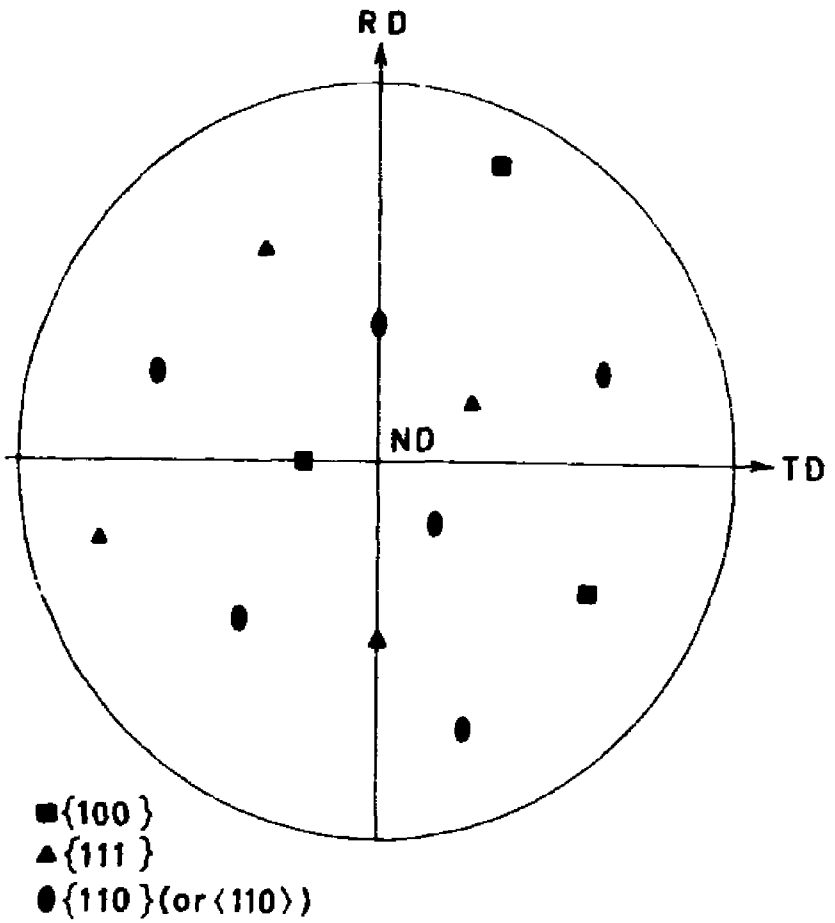

Fig. 5. Stereographic projection of crystal with slip direction and slip-plane normal lying in the plane determined by RD and ND.

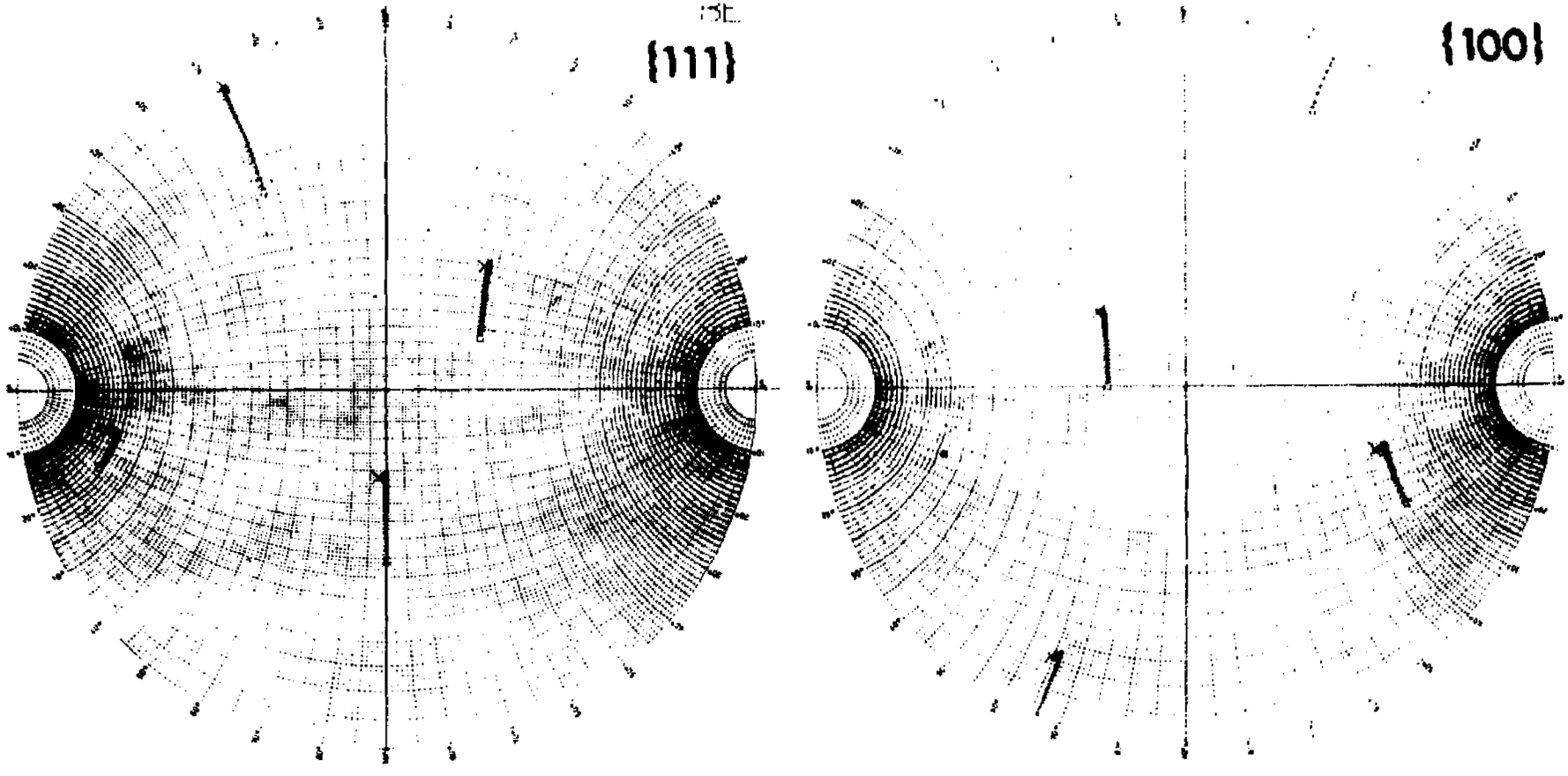

Fig. 6. The lattice rotation of the crystal from fig. 5 represented by the rotation path of the {111} and {100} poles plotted for each shear step of 0.05 during 45% reduction in ROLTEX I with only the basic stresses applied. A Wulff net is superimposed. The initial lattice rotation is a simple rotation about TD.

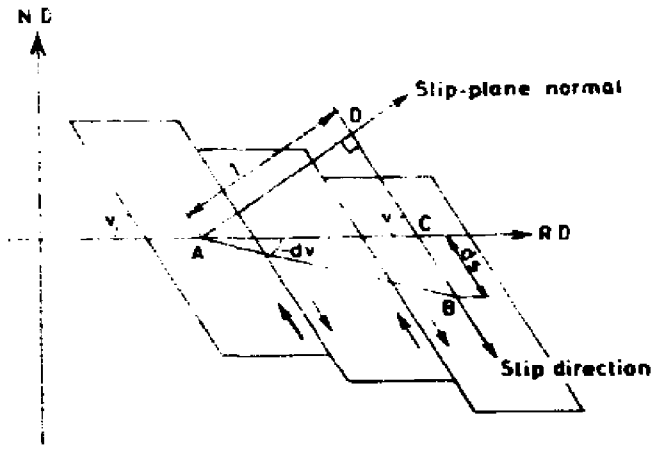

Fig. 7. Section in the crystal from figs. 5 and 6 while it is gliding with slip direction and slip-plane normal in the plane determined by RD and ND, i. e. while the lattice rotation is "simple".

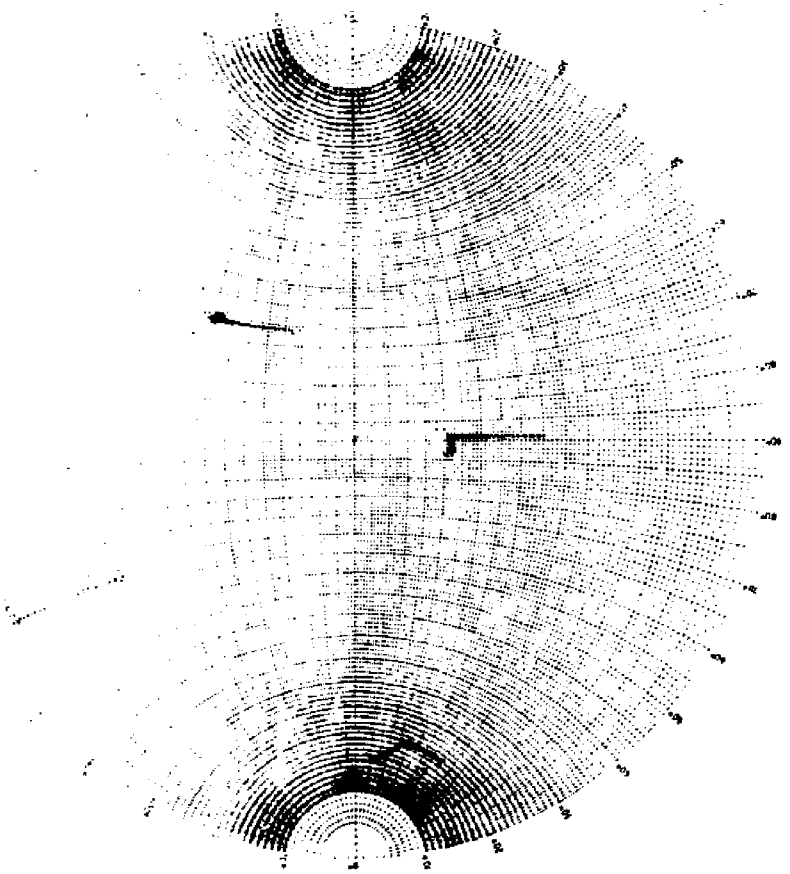

Fig. 8. The rotation path of the {111} poles of the crystal from fig. 5 rotated 90° about ND. The deformation takes place with a special basic stress system consisting of a tensile stress along TD and a compressive stress along ND. The initial rotation path is identical to that in fig. 6 rotated 90° about ND.

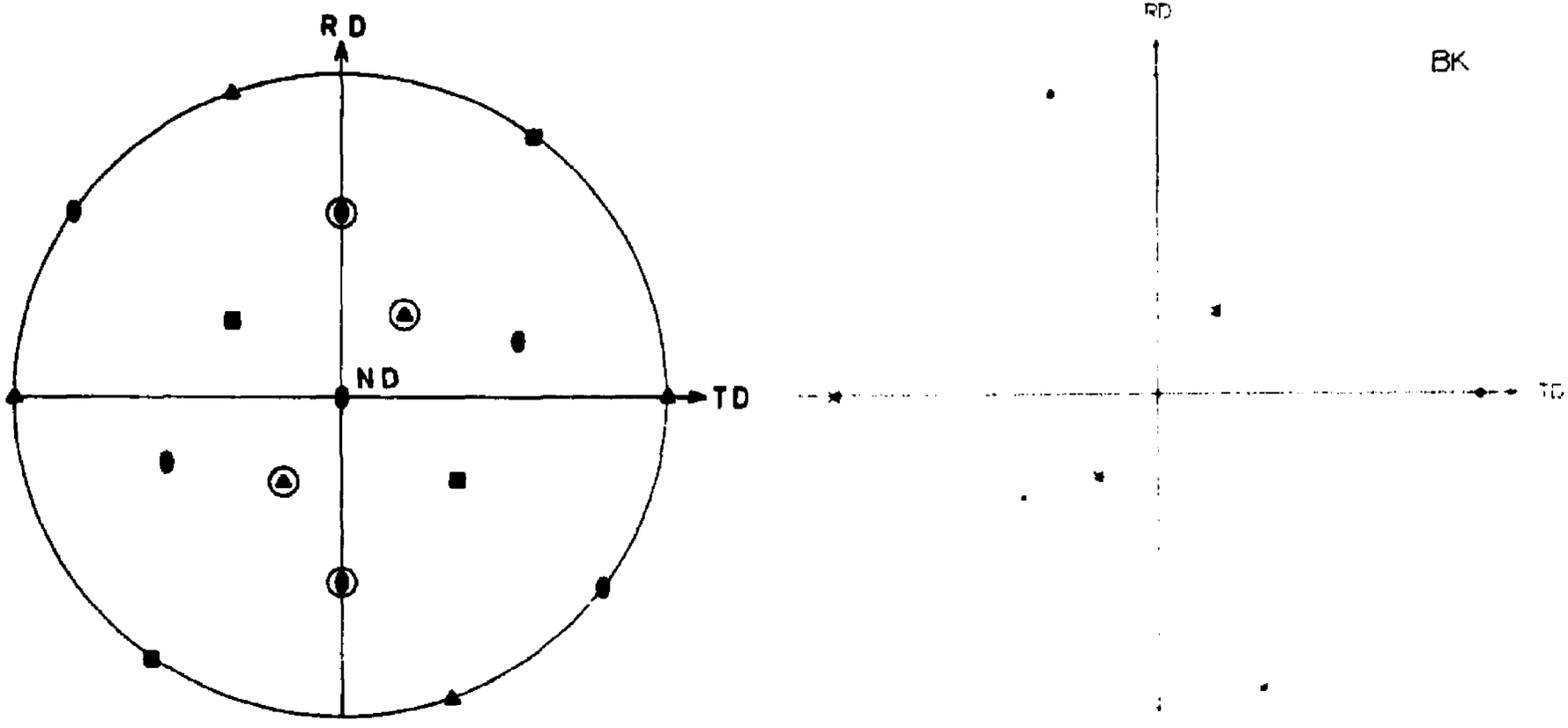

Fig. 9. The symmetrical orientation {110} ⟨211⟩. (a) stereographic projection with operative slip planes and slip directions marked with circles. (b) rotation path or more correctly orientation oscillation produced by 25% reduction with ROLTEX 1 plotting the {111} poles for each shear step of 0.05. Only the basic stress system is applied.

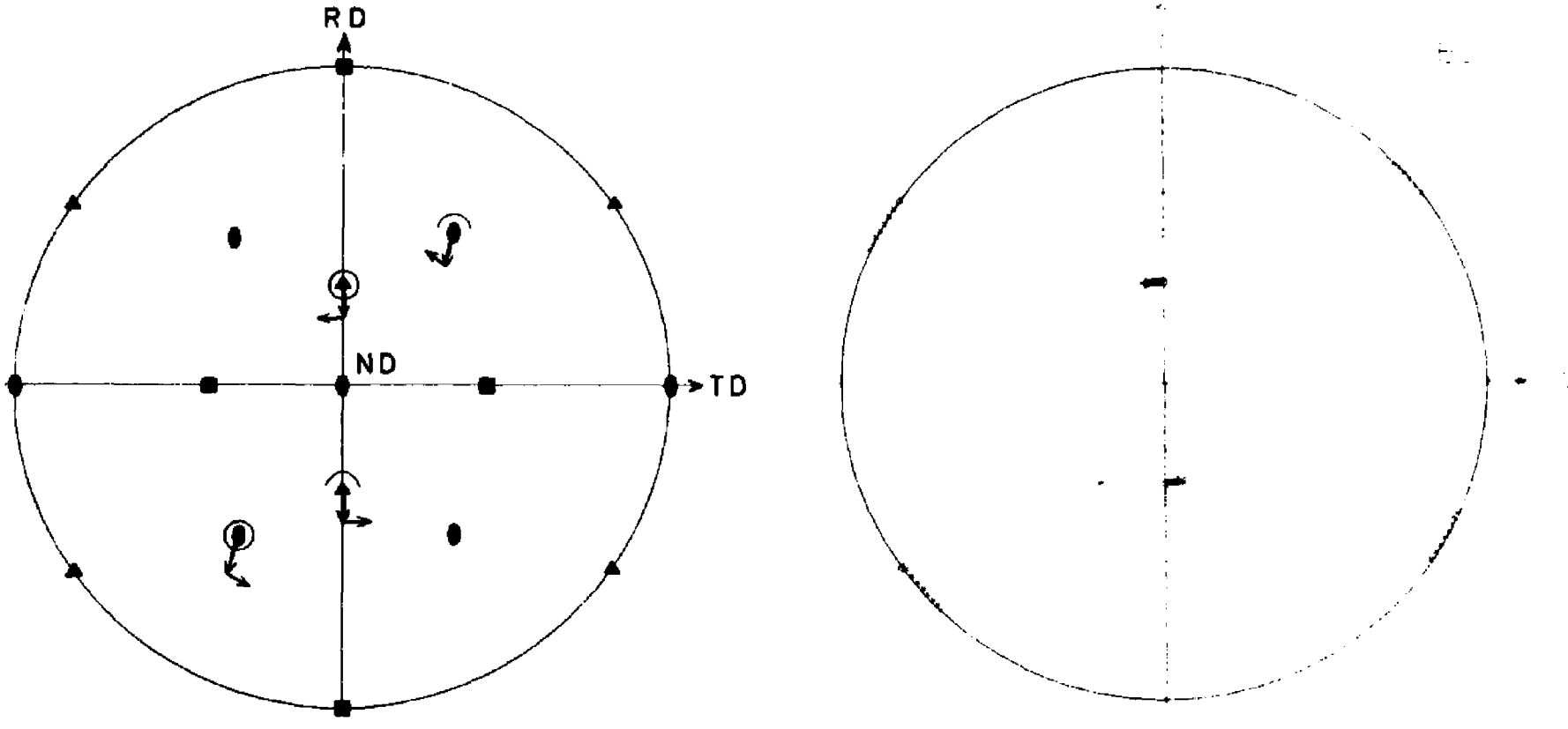

Fig. 10. Another symmetrical orientation, {110} ⟨100⟩. (a) stereographic projection with the two active slip systems marked with full and half circles and with the approximate rotation path resulting from the full-circle system indicated. (b) rotation path of the {111} poles for 25% reduction in ROLTEX 1 in steps of 0.05 with basic stresses only. The lattice rotation is a simple rotation about ND.

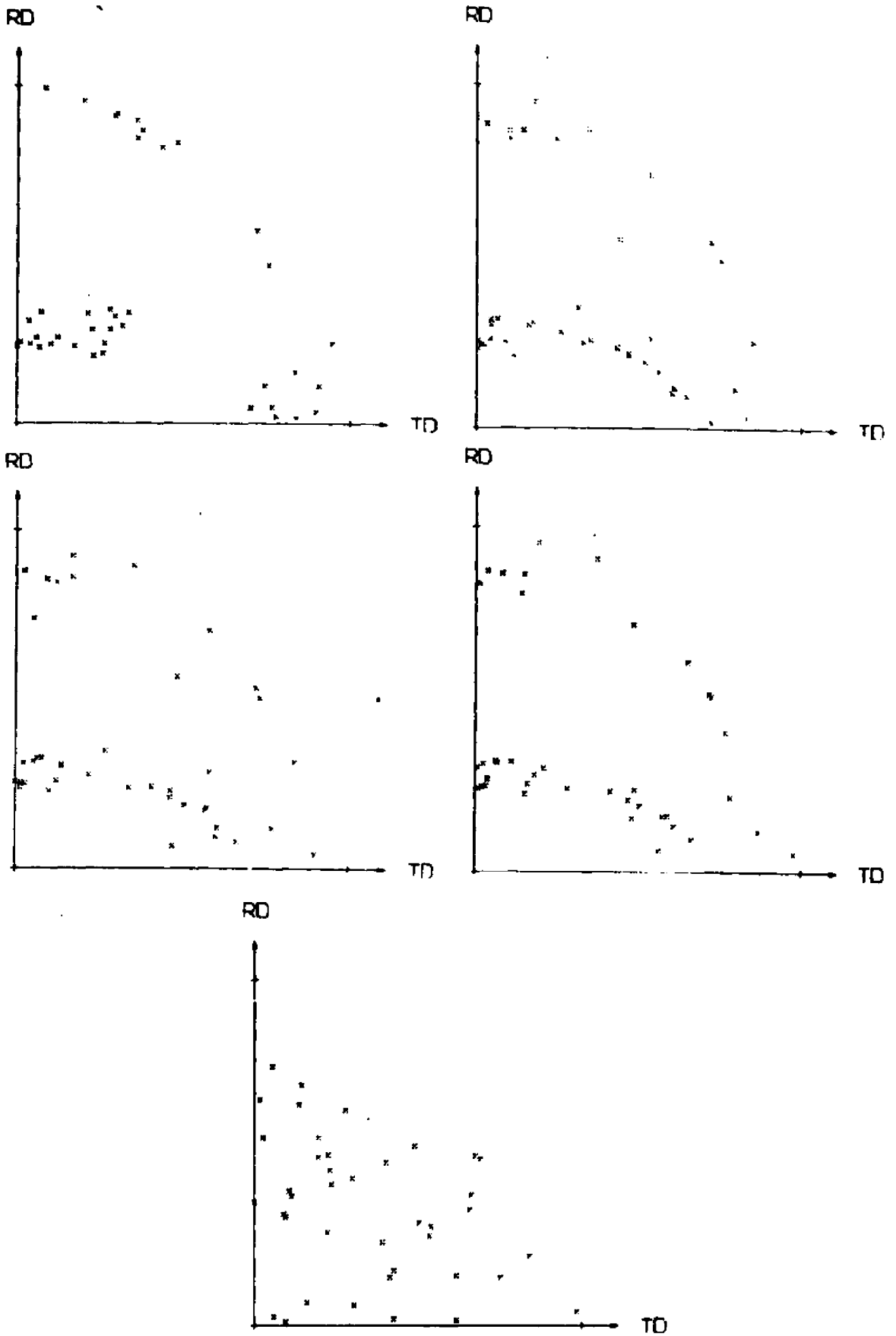

Fig. 11. The final {111} poles of the same 10 crystals of initially random orientation deformed 60% in shear steps of 0.05 with different values of the K-parameters (all the parameters K, Ks12, Ks13, Ks23 are equal in each example). (a) K = 0. (b) K = 10. (c) K = 25. (d) K = 50. (e) K = 200, a reduction of 60% is not reached as shown in table II.

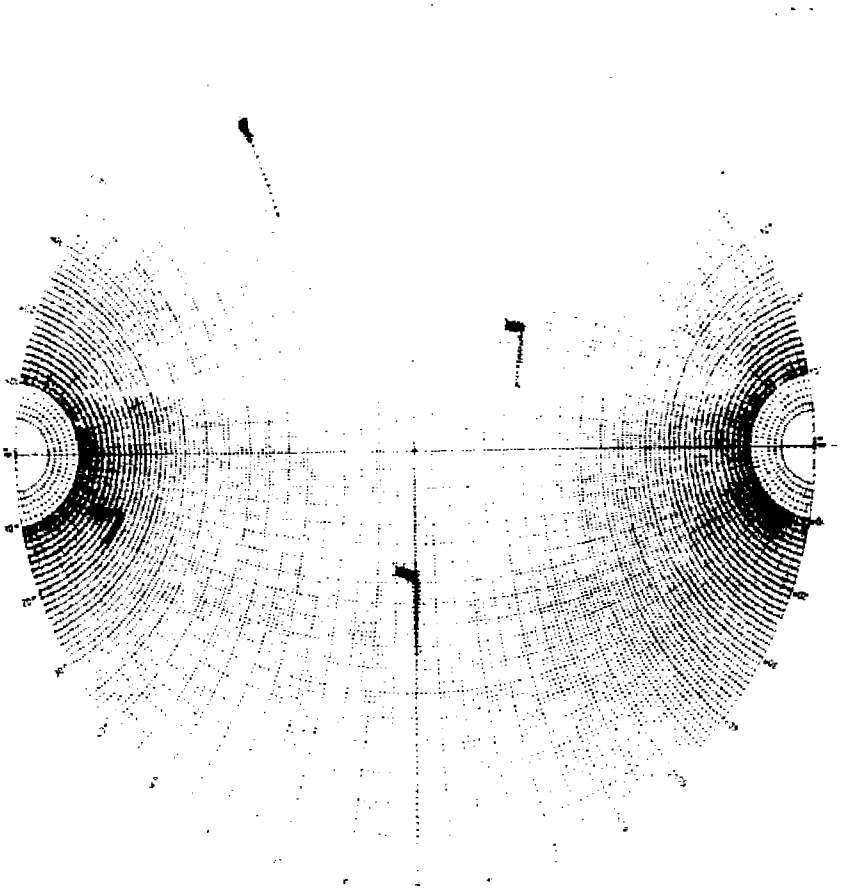

Fig. 12. The lattice rotation of the crystal from fig. 5 deformed to 50% reduction with additional stresses, illustrated by the {111} poles. Initially it is a simple rotation about TD as it is without additional stresses (fig. 6), but the simple rotation path is shorter than that in fig. 6.

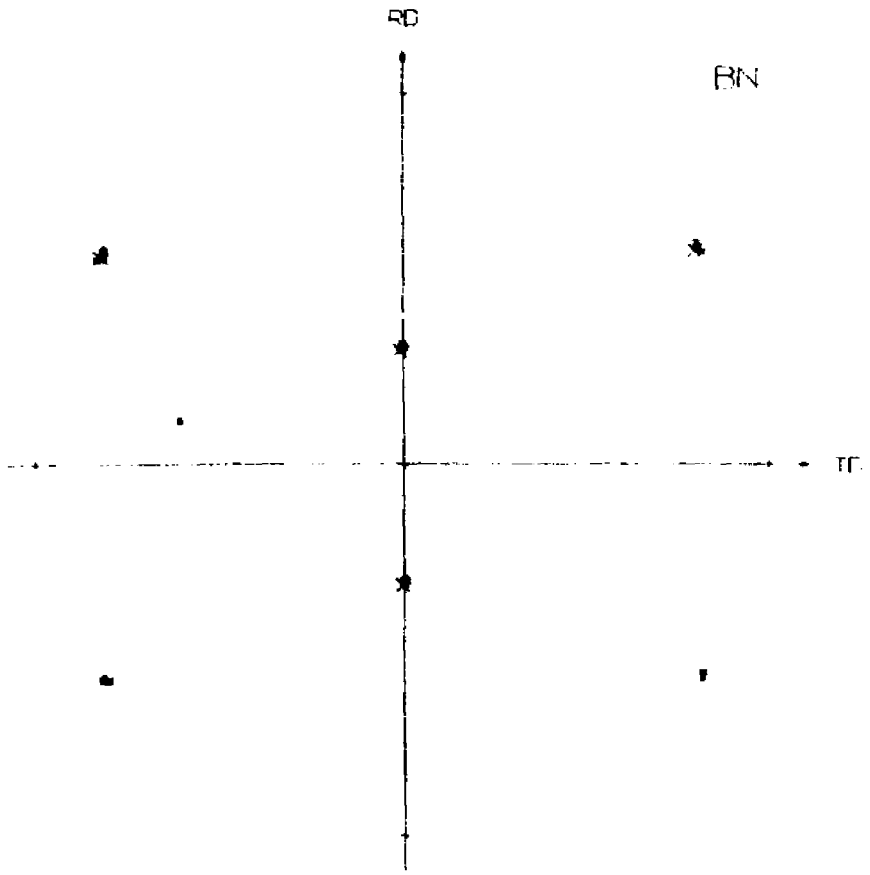

Fig. 13. The {111} poles of a {110}⟨100⟩ crystal (see fig. 10a) plotted for each shear step of 0.05 during deformation to 25% reduction in ROLTEX 1 with additional stresses applied (K = Ks = 20). The orientation is stable.

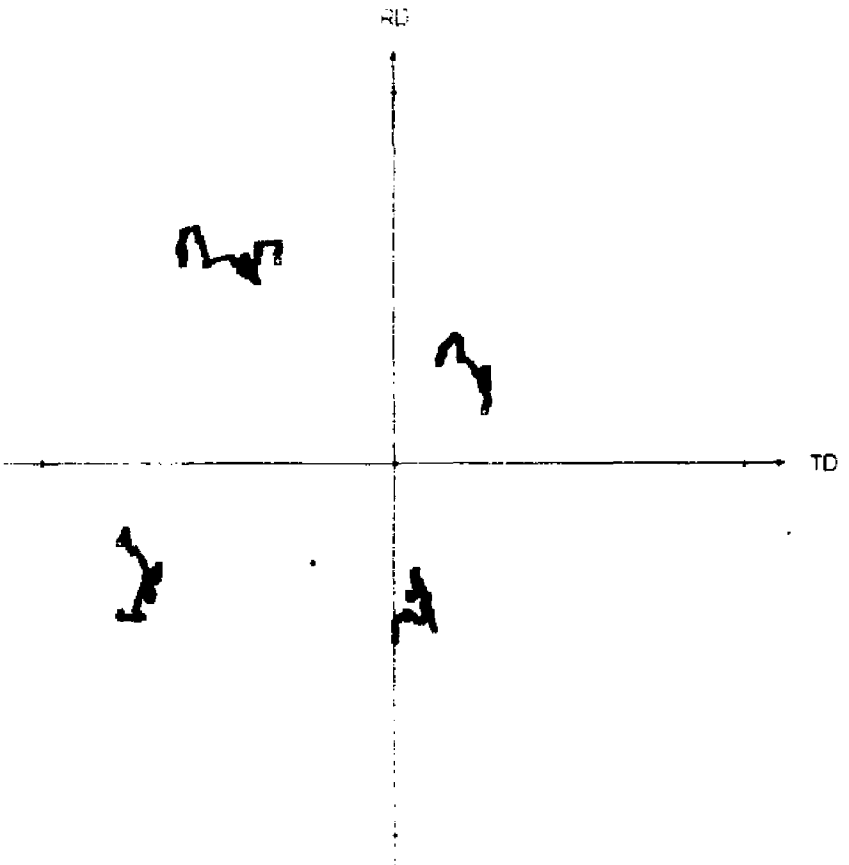

Fig. 14. The crystal from fig. 5 deformed to 50% reduction with a high random-stress level. The {111} poles shown seem to perform a "random walk".

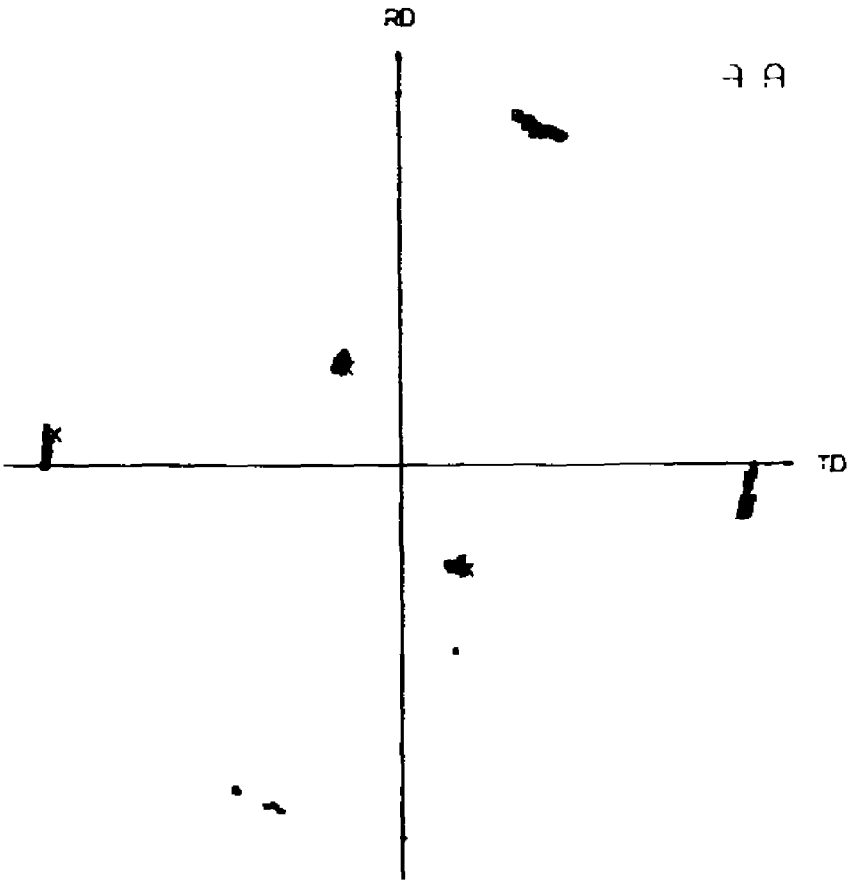

Fig. 15. A crystal with {110}⟨211⟩ orientation (corresponding to fig. 9a) is deformed in 75 steps of 0.025 shear with the average numerical random stress component 0.5. The orientation is approximately stable, but the random stresses produce fluctuations.

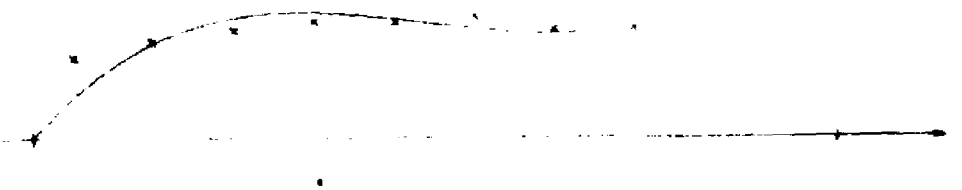

Fig. 16. HORS dOEUVRE plot for K = Ks = 20 (corresponding to the output in Appendix E). The average of the numerical value of all the additional stress components is plotted versus reduction and approximated with a third-degree polynomial.

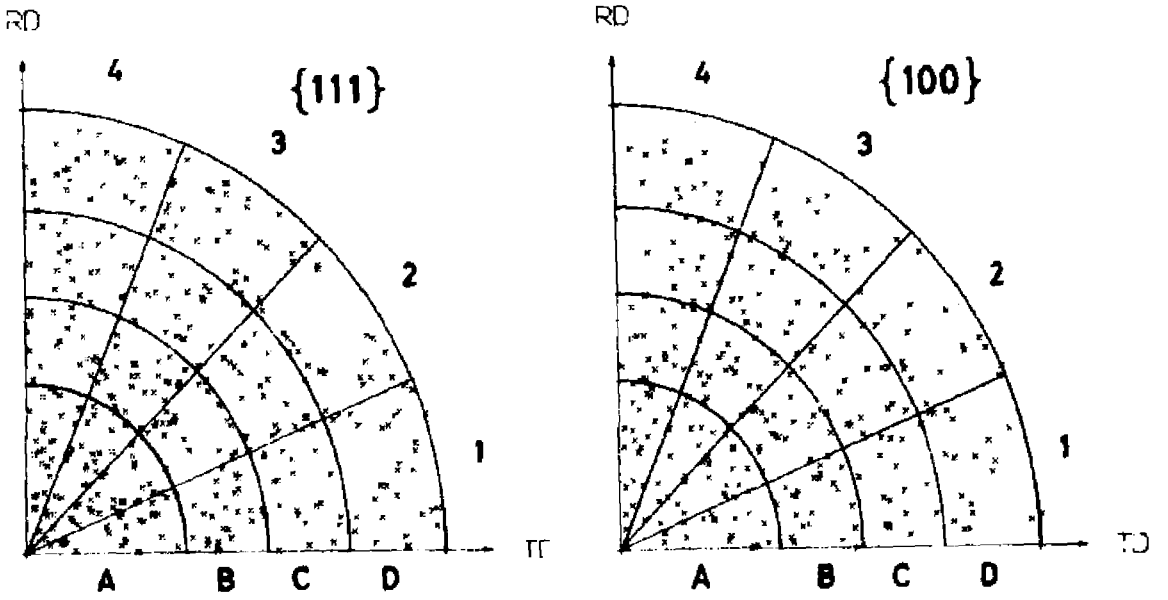

Fig. 17. The distribution of {111} and {100} poles for 100 computer-generated crystals (indicator = 1, i.e. no deformation and plotting in one quadrant only). The $\pi/2$ solid angle is divided into 16 areas each covering the same solid angle. The distributions are seen to be reasonably even, cf. table III.

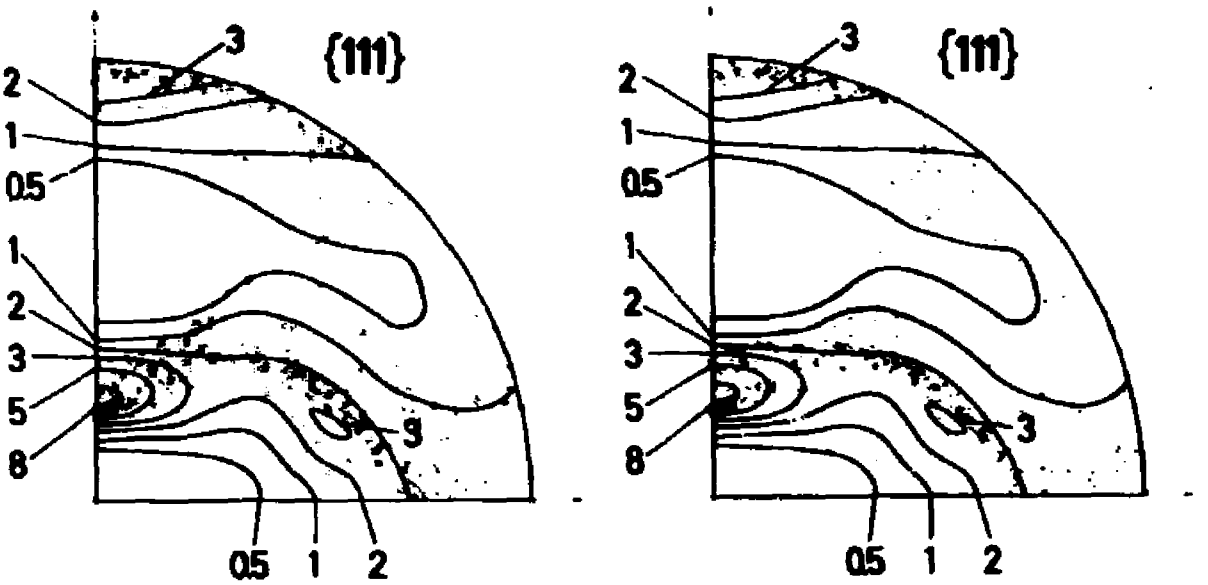

Fig. 18. Two {111} pole figures calculated with ROLTEX 2 for different values of crystalparameter (K = Ks23 = 0, Ks12 = Ks13 = 20, 50% reduction, no random stresses). Experimental copper pole figures are superimposed. There is no systematic difference between the two pole figures.

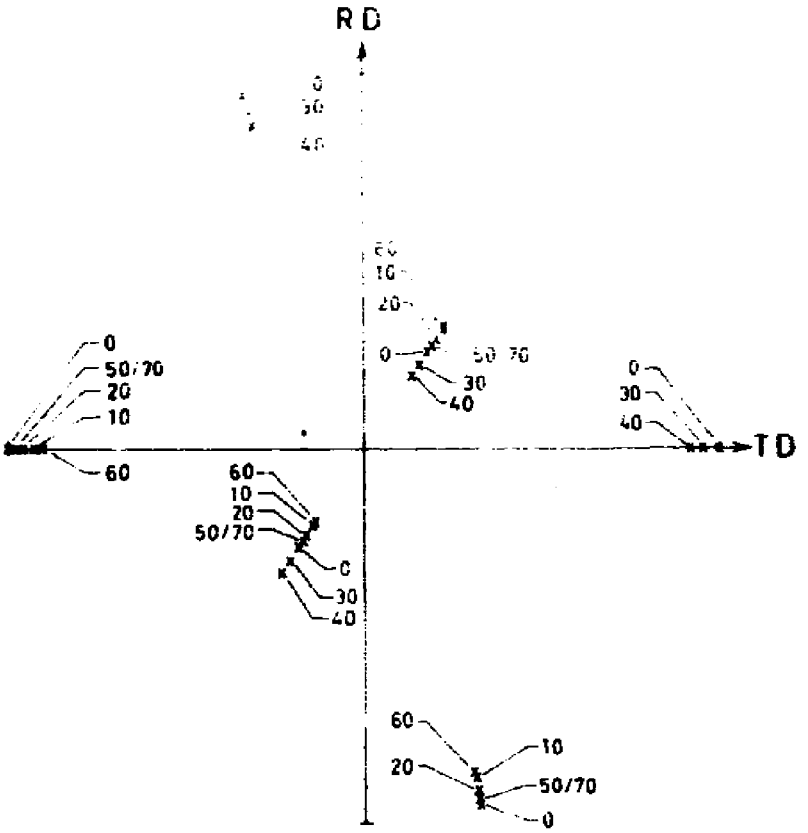

Fig. 19. The {111} poles of the crystal from fig. 9a deformed in ROLTEX 2 with SHOOT = 1.33 (no additional stresses, no random stresses). The figure presents the results of seven problems performing 10, 20, 30, 40, 50, 60, and 70 steps, i.e. it is not a direct computer plot. The {110} ⟨211⟩ orientation is not stable with overshooting in the sense that it was without overshooting (fig. 9b).

```
begin comment: A.E.K. - ASM 2B - July 1965;
integer pagecounter, linecounter, problem no,day,month,year,drum,a,
        xxx,yyy,pl111,l111,plabc;
real deltax,deltay;

procedure head; CR(100);

procedure stop;
begin integer a;
writetext(‡<
stop‡);
a := typechar;
if a > 128 then a := a - 128;
if a = 50 then go to start;
if a = 53 then go to end of program;
end of stop;

procedure CR(a);
value a; integer a;

begin
if linecounter - 6 < a then a := linecounter + 2;
linecounter := linecounter - a;
for a := a - 1 step -1 until 0 do outchar(64);

if linecounter < 0 then begin
pagecounter := pagecounter + 1;
linecounter := linecounter + 64;
if pagecounter > 1 then
begin outsp(32); output(‡-ddd‡, -pagecounter, outtext(‡<-‡)) end;
outtext(‡<

A.E.K. - Program nr. 431 - Oppave nr.‡);
output(‡ndd‡, problem no);
outtext (‡< - ‡);
output(‡nd‡, day, outtext (‡<.‡), month); outtext(‡<.‡); output(‡nddd‡,year); .

comment NOW ONE LINE TO BE PRINTED IN EACH HEADING CAN BE WRITTEN;

outtext(‡<
ROLTEX 2


‡)
end of linecounter <0
end of CR;

procedure plotline(x0,y0,x1,y1);
value x0, y0,x1,y1; real x0,y0,x1,y1;
begin integer xx,yy,tr; boolean hop; boolean array B[1:1111];
comment SA 74;
tr := drumplace; drumplace := pl111;
from drum(B); drumplace := tr;
xx := x0/deltax-xxx; yy := y0/deltay-yyy;
gierprog(B[2],hop,xx,yy);
xxx := xxx + xx; yyy := yyy + yy;
xx := x1/deltax - xxx; yy := y1/deltay - yyy;
gier(hop);
xxx := xxx + xx; yyy := yyy + yy
end plotline;
```

```
real procedure plotsymbol(t,x,y,h);
value t,x,y,h; integer t; real x,y,h;
begin integer xx,yy,n1,n2,hh,i,x1,y1,samk; boolean hop, AA;
boolean array A[0:1],B[1:1111];
comment SA 77/1;
hh := h * 20;
xx := drumplace; drumplace := pl111;
from drum(B); drumplace := plabc - t * 2;
from drum(A); drumplace := xx;
AA := A[0];
x1 := split(AA,0,0,samk,1,4,n1,5,9,y1) : 6;
y1 := (y1-x1*6) * hh; x1 := x1 * hh;
xx := x1 + x/deltax - xxx; yy := y1 + y/deltay - yyy-samk*hh*2;
xxx := x/deltax; yyy := y/deltay;
gierproc(B[2],hop,xx,yy);
n2 := (n1-8) * 5; n1 := if n1 <= 7 then n1 * 5 else 35;

for i := 10 step 5 until n1, 0 step 5 until n2 do
begin
      if i = 0 then AA := A[1];
      xx := split(AA,i,i+4,yy) : 6;
      yy := (yy-xx*6) * hh-y1;
      xx := xx * hh - x1;
      gier(hop);
      x1 := x1 + xx; y1 := y1 + yy end i;
xxx := xxx + x1; yyy := yyy + y1-samk*hh*2;
plotsymbol := x + split(AA,35,39,yy)/.3* h * deltax
end plotsymbol;


plabc :=plam12;l111:=40;pl111:=plabc-160;
xxx:=yyy:=0;deltax:=deltay:=1;
drum:=drumplace;
linecounter:=0;
writetext(‡<plotter‡);typechar;
for a:=1 step 1 until 30 do outchar(112);outclear;


start:
drumplace := drum;
pagecounter := 0;
problem no := inone;
if problem no  < 0 then go to end of program;

input(day,month,year);
for a := 1 step 1 until 30 do outchar(112);
head;
if year < 1965 ∨ year > 1970 then begin
writetext(‡<
aar‡); write(‡-nddd-dd‡,year); writetext(‡< i opg.‡); write(‡ndd‡,problem no);
stop;
end of test;
comment NOW COMES THE PROGRAM;
comment  NOLTEX 2, P. 431 A.E.K. 20 juni 1967;

begin
integer indicator, i, RANDOM, STOP, randomparameter,
        crystalparameter, k, counter,p,q,r,s,t,rr,ø,P,Q,PP,QQ,RQ;
integer array dr[1:2];
begin
real K,Ks12,Ks13,Ks23,L,SHOOT,R,step,sq,u,red,length,thickness,transdim,SA,
     fault,A11,A12,A13,A21,A22,A23,A31,A32,A33, LL1,LL2,LL3;
boolean LLpos;
real array f,fr[1:3,1:3],n,F[1:4,1:3],d[1:4,1:3,1:3];
```

```
integer array rcounter, RC[1:3,1:5];

real procedure random (A,B);
real A,B;
begin randomparameter:= randomparameter*125;
      randomparameter:= randomparameter- randomparameter:2796203*2796203;
      random:= randomparameter* (B-A)/2796203+A
end random;

real procedure randomcryst(A,B);
real A,B;
begin crystalparameter:=crystalparameter*125;
      crystalparameter:=crystalparameter- crystalparameter:2796203*2796203;
      randomcryst:= crystalparameter* (B-A)/2796203+A
end randomcryst;

begin comment coordinate systems;
real cx,cy;
real procedure plottext(string,x,y,h,d);
value h,d,string; real x,y,h,d; boolean string;
begin boolean array b[0:39];
integer i,j,tp,t,case; real x0;
x0 := x; case := 0;
if split(string,0,3,t) = 10 then begin i := 40;
for i := i-6 while split(string,i,i+5,t)=10
do if t = 63 then begin x := x0; y := y-d * h end else begin
if t = 58 then case := 0
else if t = 60 then
case := 128 else x:= plotsymbol(t+case,x,y,h) end end else
begin
tp := drumplace;
split(string,10,19,1,30,39,j);
drumplace := 40*j+39; from drum(b); j := 42;
for j := j - 6 while split(b[i],j,j+5,t) = 10 do begin
if t = 63 then begin x := x0; y := y-d*h end else
begin if t = 58 then case := 0
else if t = 60 then case := 128 else
x := plotsymbol(t+case,x,y,h) end;
if j = 0 then begin j := 42; i := i+1 end;
if i > 39 then begin i := 0; from drum(b) end
end;
drumplace := tp end;
plottext:=x;
end plottext;

procedure plotaxes(x0,y0,xmin,xmax,ymin,ymax,xmark,ymark);
value x0,y0,xmin,xmax,ymin,ymax,xmark,ymark;
real x0,y0,xmin,xmax,ymin,ymax,xmark,ymark;
begin real y1,y2;
integer xx,yy,tr,j;
boolean hop,plus,x;
boolean array B[1:1111];
procedure pil;
begin xx := -5; yy := -20; gier(hop);
xx := 10; yy := 0; gier(hop);
xx := - 5;yy := 20; gier(hop);
end;
tr := drumplace; drumplace := pl 111;
from drum(B); drumplace := tr;
plus := abs(yyy*deltay -ymin) > abs(yyy*deltay - ymax);
x := false;
for j := 1, 2 do begin
y1 := if plus then ymax else ymin;
xx := x0/deltax-xxx; yy := y1/deltay-yyy;
if x then gierproc(B[2],hop,yy,xx) else gierproc(B[2],hop,xx,yy);
xxx := xxx + xx; yyy := yyy + yy;
if plus then pil;
```

```
for y2 := y0 + entier((y1-y0)/ymark+(if plus then -.01 else 1.01)) x ymark
        while y2 < ymax ∧ y2 ≥ ymin
do begin
xx := 0; yy := y2/deltay - yyy;
gier(hop);
yyy := yyy + yy;
xx := 10; yy := 0; gier(hop); xx := -20; gier(hop); xx := 10; gier(hop);
y1 := y2; end;
xx := 0; yy:=(if plus then ymin else ymax)/deltay - yyy;
gier(hop); yyy:= yyy+yy;
if -. plus then pil;
x:= plus := true; ymin := xmin; ymax := xmax;
y1 := ymark; ymark:= xmark; xmark:= y1;
y1 := y0; y0 := x0; x0 := y1; y1 := xxx; xxx := yyy; yyy := y1;
y1 := deltax; deltax := deltay; deltay := y1; end;
end;



real procedure plotnumber(layout,tal,x,y,h);
value tal,x,y,h;
string layout; real tal,x,y,h;

begin comment A.E.K.-SA 91, april 1967.Description in SA 91;
integer ans,int,magn,exp,ti,zero,spaces,førp,eftp,fn,n,fe,inexp;

procedure anslag(symbol); integer symbol;
begin
      if symbol=0 then x:=x+h else x:=plotsymbol(symbol,x,y,h);
      if ans>1 then ans:=ans/2;
      if spaces> ans then begin spaces:=spaces-ans;x:=x+h end
end anslag;

procedure plotint(int,length,n,fn,pointplace);
value  int,length,n,fn,pointplace;
integer int,length,n,fn,pointplace;
begin integer ciffer,q;boolean signif;

if fn=3 then begin x:=plotsymbol(if int<0 then 32 else 160,x,y,h);fn:=0 end

       else fn:= if int<0 then 32 else if fn=2 then 160 else fn=128;
if fn=128 then begin x:=x+h;fn:=0 end;
int:=abs(int); signif:=false;q:=ti↑length;

for length:= length-1 while length ≥ 0 do
begin
     q:=q/ti;ciffer:=int:q;int:=int-ciffer×q;
     if -.signif then signif:=ciffer≠0 ∨ length=pointplace-1+n;
     if  signif then begin if ciffer=0 then ciffer:=16;
                           if     fn≠0 then begin x:=plotsymbol(fn,x,y,h);fn:=0 end;
                           if length=pointplace-1 then anslag(59)
                     end;
     anslag(ciffer)
end whilestep;
if fn≠0 then x:=x+h
end plotint;

zero:= -split(layout,0,19,spaces,28,29,fn,34,34,n,24,27,førp,30,33,eftp,
38,39,fe,35,37,inexp,20,23,zero)+ førp+eftp;
if fn=1 ∧ eftp=0 then n:=1;

exp:=0;ti:=10;if tal<0 then begin fn:=fn-ti;tal:=-tal end;
ans:=-ti↑inexp;magn:=ti↑(førp-1);
```

```
comment venstreforskydning;
if fe=0 then exp:=-1
        else
for exp:=exp-1 while exp>-1 and abs(tal)<magn do tal:= tal*ti;

comment højreforskydning;
magn:= if tal=0 then -1 else entier(ln(tal)/2.3025);
for exp:=exp+1 while magn>førp do begin tal:= tal/ti;magn:=magn-1end;

int:=entier(tal*ti/eftp+.5);
if int=0 then zero:=exp:=0
        else
begin
      if int>= ti/(førp+eftp) then begin exp:=exp+1;int:=int/ti end;
      if zero<>0 then
begin
      zero:=zero+1;
      if inexp <exp=0then magn:= 0
                    else begin magn:=entier(exp/zero+.99)*zero-exp;exp:=exp+magn end;
      zero:=zero+entier(ln(int)/2.3025)-førp-eftp-magn;
      if zero<= 0 then int:=entier(int/ti/magn+.5)
                  else
      begin int:=entier(int/ti/(magn+zero)+.5);
            if eftp>zero then eftp:=eftp-zero
                         else
      begin if exp=0 then begin int:=int*ti/(zero-eftp);zero:=eftp;eftp:=0 end
                         else eftp:=eftp-zero;
            if eftp+zero<>0 then zero:=zero+1
      end
      end
end betydende cifre
end int<>0;

if fn<0 then begin int:=-int;fn:=fn+ti end;
ans:= 524 288;
plotint(int,førp+eftp,n,fn,eftp);
for zero:=zero-1 while zero>0 do anslag(0);

if exp=0 then for inexp:=inexp-1 while inexp> 0 do
              begin anslag(0);if inexp=0 then x:=x+h;
                    if fe<>0 then begin x:=x+h;fe:=0 end
              end
         else begin if exp> ti/inexp then inexp:=entier(ln(exp)/2.3025)
                                     else
                    for inexp:=inexp-1 while fe<>3*ti/inexp> abs(exp) do anslag(0);
                    x:=plotsymbol(155,x,y,h);
                    plotint(exp.inexp+1,0,fe,0)
              end;
plotnumber:=x
end plotnumber;

comment now coordinate systems and input;
CR(2); outtext (<CONSTANTS: >);  CR(2);
cx:=2420;cy:=1280;
plottext(<TD>, cx  ,cy,0.4,2);
cx:=100; cy:=2350;
sq:= plottext(<100 FOR PROBLEM >, cx  ,cy,0.4,2);
plotnumber (<ndd>, problem no,sq,cy,0.4);
cx:=1218; cy:=2470;
plottext(<RD>,cx,cy,0.4,2);
input (indicator);  if  indicator > 0.5 then
plotaxes(1250,1300,1250,2350,1300,2400,990,990) else
plotaxes(1250,1300,150,2350,200,2400,990,990);
```

```
cx:=2420; cy:=3680;
plottext(‹<TD›,cx,cy,0.4,2);
cx:=100; cy:=4750;
sq:=plottext(‹<111 FOR PROBLEM ›, cx,cy,0.4,2);
plotnumber(‹ndd›,problem no,sq,cy,0.4);
cx:=1218; cy:=4870;
plottext(‹<RD›,cx,cy,0.4,2);
if indicator > 0.5 then
plotaxes(1250,3700,1250,2350,3700,4800,990,990) else
plotaxes (1250,3700,150,2350,2600,4800,990,990);
end block with plottext, -axis and -number;

input ( K,Ks12,Ks13,Ks23,SHOOT,R,step,STOP,1,RANDOM, randomparameter,
      crystalparameter, LL1,LL2,LL3);
outtext (‹<     indicator: ›); output(‹-n›,indicator); CR(1);
outtext (‹<shape force factors: ›);
output (‹ndd.d›,K,outsp(2),Ks12,outsp(2),Ks13,outsp(2),Ks23);
outtext (‹<     overshooting factor: ›);
output (‹n.ddd›, SHOOT); CR(1); outtext(‹<reduction in percent: ›);
output(‹nd.d›,R); outtext(‹<     step size:   ›); output (‹n.ddd›,step); CR(1);
outtext(‹<maximum number  of steps: ›); output (‹ndd›, STOP); outtext
(‹<     number of crystals: ›); output (‹ndd›,1); CR(1); outtext(‹<RANDOM ›);
output (‹ndd›,RANDOM); outtext(‹<     randomparameter: ›); output (‹ddddddd›,
randomparameter); CR(1);
outtext (‹<crystalparameter: ›); output(‹ddddddd›,crystalparameter); CR(1);
outtext(‹<LL[1-3]: ›); output(‹-n.ddd-d›,LL1,outsp(3),LL2,outsp(3),LL3);
CR(3); outtext(‹<RESULTS: ›); CR(1); outtext (‹<steps/fault›); CR(2);
        dr[1]:= drumplace;
        dr[2]:= drumplace-8x1-400;
        t:= 0;
        LLpos:= LL1>-0.5 ∨ LL2>-0.5 ∨ LL3>-0.5;
comment now the crystals one by one;
for k:= 1 step 1 until 1 do
begin
if indicator > -0.5 then
begin
A13:= randomcryst(0,1);
if A13 < 0.99999 then
begin
     u:= randomcryst (0,1);
     u:= randomcryst (0,1.570796);
     sq:= sqrt(1-A13↑2);
     A11:= cos(u)×sq;
     A12:= sin(u)×sq
end else
begin
      A11:= A12:= 0;
      A13:= 1;
end;
     u:= randomcryst (0,1);
     u:= randomcryst (0,3.141593);
if A12 ↑ 2 + A13 ↑ 2 < 0.99998 then
begin
     A23:= sin (u) × sqrt (A11 ↑ 2 + A12 ↑ 2);
     sq:= A12 × A13/sqrt ((A11 ↑2 +A12 ↑2) × (A11 ↑ 2 + A13 ↑ 2));
     u:= u - arctan (sqrt(1-sq↑2)/sq);
     A22:= - sin (u) × sqrt (A11 ↑ 2 +A13 ↑ 2);
     A21:= - (A12 × A22 + A13 × A23)/A11;
end else
begin
     A11:= 0;
     sq:= sqrt (A12 ↑ 2 + A13 ↑ 2);
     A12:= A12/sq;
     A13:= A13/sq;
     A21:= sin (u);
     sq:= cos(u);
```

```
    A22 := sqx A13;
    A23 :=-sqx A12;
end;
if abs (1-sqrt (A21 ↑2 + A22 ↑ 2 + A23 ↑ 2)) > 3w-4 then
outtext (‡<-y   ‡);
end else
begin
    input(A11,A12,A13,A22,A23);
    output(‡-n.dddd‡,A11,outsp(4),A12,outsp(4),A13,outsp(4),A22,outsp(4),A23);
    CR(1);
    sq := sqrt (A11 ↑2 + A12 ↑ 2 + A13 ↑ 2);
    A11 := A11/sq; A12 := A12/sq; A13 := A13/sq;
    A21 := -(A12 x A22 +A13 x A23)/A11;
     sq := sqrt (A21↑ 2 + A22 ↑2 + A23 ↑ 2);
    A21 := A21/sq; A22 := A22/sq; A23 := A23/sq;
end;

      A31 := A12 x A23 -A22 x A13;
      A32 := A13 x A21 -A23 x A11;
      A33 := A11 x A22 -A21 x A12;
if abs (sqrt (A31 ↑ 2 + A32 ↑ 2 + A33 ↑ 2 )-1) > 3w-4 then
outtext (‡<-z   ‡);
comment coordinates being generated now slip;

for q := 1,2,3 do
for r := 1,2,3 do
begin rcounter[q,r] := 0; fr[q,r] := f[q,r]:= 0;
      M.[q,r] := random(2,RANDOM);
end;
PQ:= 3; counter:= -1;
red:= 0;
      length := thickness := transdim := 1;

      n[1,1] :=n[2,1] :=n[3,1] :=n[4,1] :=n[1,2] :=n[2,2] :=n[1,3] :=n[3,3] :=sqrt(1/3);
      n[3,2] :=n[4,2] :=n[2,3] :=n[4,3] :=-n[1,1];
      d[1,1,1] :=d[1,2,1] :=d[2,1,1] :=d[2,2,1] :=d[3,1,1] :=d[3,2,1] :=d[4,1,1] :=
           d[4,2,1] :=d[1,3,2] :=d[2,3,2] :=d[3,1,2] :=d[3,3,2] :=d[4,1,2] :=d[4,3,2] :=
           d[2,2,3] :=d[2,3,3] :=d[3,3,3] :=d[4,2,3] :=sqrt(1/2);
      d[1,1,2] :=d[2,1,2] :=d[1,2,3] :=d[1,3,3] :=d[3,2,3] :=d[4,3,3] :=-d[1,1,1];
      d[1,3,1] :=d[2,3,1] :=d[3,3,1] :=d[4,3,1] :=d[1,2,2] :=d[2,2,2] :=d[3,2,2] :=
           d[4,2,2] :=d[1,1,3] :=d[2,1,3] :=d[3,1,3] :=d[4,1,3] := 0;
new slip event:
      counter := counter +1;
begin real np1,np2,np3, dpq1,dpq2,dpq3;
for p:= 1,2,3,4 do
begin
    np1 := n[p,1]; np2 := n[p,2]; np3 := n[p,3];
    n[p,1]:= np1xA11+ np2xA21+ np3xA31;
    n[p,2]:= np1xA12+ np2xA22+ np3xA32;
    n[p,3]:= np1xA13+ np2xA23+ np3xA33;
for q := 1,2,3 do
begin dpq1 := d[p,q,1]; dpq2 := d[p,q,2]; dpq3 := d[p,q,3];
    d[p,q,1]:= dpq1xA11+ dpq2xA21+ dpq3xA31;
    d[p,q,2]:= dpq1xA12+ dpq2xA22+ dpq3xA32;
     d[p,q,3]:= dpq1xA13+ dpq2xA23+ dpq3xA33
end q
end p end block;
if abs (indicator) < 1.5 ∨ red > R/100 ∨ counter ≥ STOP then go to plot;

SA := 0; L:= abs(((LL3xred +LL2)xred +LL1)xred);
begin real Fpq,Fp1,Fp2,Fp3, Spq;
for p := 1,2,3,4 do
begin
      F[p,1] := 0;
      F[p,2] := n[p,2];
```

```
      F[p,3] := -n[p,3];
for q := 1,2,3 do
begin
      Fpq:= F[p,q];
for r := 1,2,3 do
begin
      Fpq := Fpq + n[p,r] * (fr[r,q] + fr[r,q]);
if LLpos^ p=4 then
begin
       rr:= rcounter[q,r] := rcounter[q,r] + 1;
      if rr= RC[q,r] then
      begin
            rcounter[q,r]:=0;
            if q > r then
            begin fr[q,r] := fr[r,q]; RC[q,r] := RC[r,q]
            end
            else
            begin RC[q,r] := random(1,RANDOM);
                  fr[q,r]:=  random (-L,L);
            end
      end
end LLpos^p=4
end r-step;
      F[p,q]:= Fpq
end q-step;

Fp1:= F[p,1]; Fp2:= F[p,2]; Fp3:= F[p,3];
for q := 1,2,3 do
begin
      Spq := Fp1 * d[p,q,1] + Fp2 * d[p,q,2] + Fp3 * d[p,q,3];
if SHOOT > -0.5 ^ p= PP ^ q = QQ ^ PQ > 3.5 then
        Spq:= Spq * SHOOT;
if abs(Spq) > SA then
begin
      SA := abs(Spq);
      P := p; Q := q;
      ø:= if Spq < 0 then -1 else 1;

end end
end p:= 1,2,3,4 end block;

begin
real d1, d2, d3, n1, n2, n3;
     d1:= d[P,Q,1]*ø;
     d2:= d[P,Q,2]*ø;
     d3:= d[P,Q,3]*ø;
     n1:= n[P,1];
     n2:= n[P,2];
     n3:= n[P,3];
     sq:= step*n2;
     A21:= -sq*d1;
     A22:= 1-sq*d2;
     A23:= -sq*d3;
     length:= (1+sq*d2)*length;
     sq:= sqrt(A21^2+A22^2+A23^2);
     A21:= A21/sq;  A22:= A22/sq;  A23:= A23/sq;
     sq:= step*n1;
     A11:= 1-sq*d1;
     A12:= step*n2*d1;
     A13:= -sq*d3;
     transdim:= (1+sq*d1)*transdim;
     sq:= sqrt(A11^2+A12^2+A13^2);
     A11:= A11/sq;  A12:= A12/sq;  A13:= A13/sq;
     A31:= A12*A23-A22*A13;
     A32:= A13*A21-A23*A11;
     A33:= A11*A22-A21*A12;
```

```
     thickness:= 1/(length*transdim);
     red:= 1-thickness;
     f[1,1]:= Ka(1-transdim);
     f[2,2]:= Ka(1-1/transdim);
     aq:= step/2;
     f[1,2]:= f[2,1]:= f[2,1]-Ka12*aq*(n1*d2+n2*d1);
     f[1,3]:= f[3,1]:= f[3,1]-Ka13*aq*(n1*d3+n3*d1);
     f[2,3]:= f[3,2]:= f[3,2]-Ka23*aq*(n2*d3+n3*d2);
end block;
if SHOUT < -0.5 then go to new slip event;
if counter < 0.5 then
begin
     PP:= P; QQ:= Q; PQ:= PQ+1
end else
if PP= P ∧ QQ=Q then
          PQ:= PQ +1
else
begin
     PP:= P; QQ:= Q;
     PQ:= 1;
end;
go to new slip event;
comment now pole figure coordinates to drum or to plotter;
plot:
begin real cx,cy,xpole,ypole;
begin
real array C00[1:8];
for p:= 1 step 1 until 4 do
begin
     xpole:= n[p,1]/(abs(n[p,3]) +1);
     ypole:= n[p,2]/(abs(n[p,3]) +1);
if indicator > 0.5 then
begin
     xpole:= abs (xpole);
     ypole:= abs( ypole);
     C00[2 xp-1] := 1238 + 990 * xpole;
     C00[2 xp]:=3688 + 990 * ypole;
end else
begin
if n[p,3] < 0 then
begin
     xpole:= - xpole;
     ypole:= - ypole;
end;
cx:= 1238 + 990 * xpole; cy:= 3688 + 990 * ypole;
     plotsymbol(130,cx,cy, 0.3);
end; end;
if indicator > 0.5 then
begin
     drumplace:=dr[1];
     to drum (C00);
     dr[1]:= drumplace;
end; end 111 pole figure;
begin
real array C00[1:6],x[1:3,1:3];
for q:= 1 step 1 until 3 do
begin
     x[1,q]:= n[1,q] + n[4,q];
     x[2,q]:= n[1,q] - n[3,q];
     x[3,q]:= n[1,q] - n[2,q];
end;
for p:= 1 step 1 until 3 do
begin
```

```
    sq:= sqrt(x[p,1]↑2 +x[p,2]↑2+x[p,3]↑2);
for q:= 1 step 1 until 3 do
      x[p,q]:= x[p,q]/sq;
if p > 2.5 then
begin
    fault:= 0;
for q:= 1 step 1 until 3 do
begin
     r:= q+1;
if r > 3.5 then
      r:= 1;
      sq:= x[q,1] × x[r,1] + x[q,2] × x[r,2] + x[q,3] × x[r,3];
if abs (sq) > fault then
      fault:= abs (sq);
end;
output(‹ndd›,counter); outsp (2); output (‹n.dddd›, fault);
if indicator > 2.5 then
begin
outsp(6); output(‹-n.ddd›, x[1,3], outsp(2), x[2,3], outsp(2),
          x[3,3], outsp(6),x[1,2], outsp(2), x[2,2], outsp(2),
          x[3,2]); CR(1);
end else
if indicator > -0.5 then
begin
    t:= t+1;
if t < 3.5 then outsp (6)
else begin
CR(1); t:= 0;
end; end else CR(1);
end;
      xpole:= x[p,1]/(abs(x[p,3])+1);
      ypole:= x[p,2]/(abs(x[p,3])+1);
if indicator < 0.5 then
begin
      xpole:= abs (xpole);
      ypole:= abs (ypole);
      C00[2xp-1]:= 1238 + 990 × xpole;
      C00[2xp]:= 1288 + 990 × ypole;
end else
begin
if x[p,3] < 0 then
begin
     xpole:= -xpole;
     ypole:= -ypole;
end;
cx:= 1238 + 990 × xpole; cy:= 1288+990 × ypole;
       plotsymbol (130,cx,cy,0,3);
end; end;
if indicator > 0.5 then
begin
         drumplace:= dr[2];
         to drum (C 00);
         dr[2]:= drumplace;
end; end 100 pole figure
end plot;
end CRYSTAL;
end CENTRAL BLOCK;

if indicator > 1.5 then
begin
comment now plot from drum;

integer near,stepnear,step,list,cx,cy,xx,yy;
boolean hop;
```

```
integer array REF[-1:149],SQU[0:5,0:5],COO[0:299];
boolean array B[1:1111];

procedure plot;
begin
REF[-1]:= SQU[p,q];
for list:= REF[-1] while list≠-1 do
begin
     r:= -1; stepnear:= 6000;
     for list:= REF[r] while list≠-1 do
     begin
          cx:= COO[2×list  ]+12 -xxx;
          cy:= COO[2×list+1]+12 -yyy;
          step:= abs(cx);if abs(cy)>step then step:= abs(cy);
          if step < stepnear then
          begin stepnear:= step; near:=r; xx:=cx; yy:=cy
          end finding near;
          r:= list
     end searching square for nearest point,
         now removal and plot of point;

     REF[near]:= REF[REF[near]];
     gierproc(B[2],hop,xx,yy);
     xxx:=xxx+xx; yyy:=yyy+yy;
     for r:= 1,-1 do
     for xx:= 6,-12,6 do
     begin yy:= rxxx; gier(hop) end cross

end plotting square
end procedure plot;

drumplace:= pl111; fromdrum(B);

for k:= 0 step 1 until 299 do COO[k]:= -1;
for drumplace:= dr[1], dr[2] do todrum(COO);
dr[1]:= dr[1]+6×i; dr[2]:= dr[2]+6×i;

t:=1258;
for s:= 3688, 1288 do
begin drumplace:= if s=3688 then dr[1] else dr[2];
              i:= if s=3688 then 6×i   else 6×i/8;
for k:= 1 step 300 until 1 do
begin
     fromdrum(COO);
     for p:= 0 step 1 until 5 do
     for q:= 0 step 1 until 5 do SQU[p,q]:= -1;

     for r:= 149, r-1 while r≥0 do
     begin
          cx:= COO[2×r  ];
          if cx > 0 then
     begin cy:= COO[2×r+1];
          p:= (cx-t):165; q:= (cy-s):165;
          if p>5 then p:=5; if q>5 then q:=5;
          REF[r]:=SQU[p,q];SQU[p,q]:= r
     end
     end determining list;

     for p:=0, p+1 while p<6 do
     begin
          for q:=5, q-1 while q≥0 do plot;
          p:=p+1;
          for q:=0, q+1 while q<6 do plot
     end plotting
```

```
end for k
end pole figures
end indicator >0.5;

plotline(0, 5500, 0, 5500);
xxx:= yyy:= 0;
end of ROLITSY;


comment THE FOLLOWING WILL BE INSERTED AFTER THE PROGRAM;
go to start;

end of program:
for a:=1 step 1 until linecounter do outer;
outsum; for a := 1 step 1 until 60 do outchar(112);
end;
```

Appendix B

A.E.K. - Program nr. 431 - Opgave nr. 49 - 10. 4.1968
ROLTEX 2

CONSTANTS:

Indicator: 2
shape force factors: 0.0 20.0 20.0 0.0 overshooting factor: -1.000
reduction in percent: 80.0 step size: 0.050
maximum number of steps: 150 number of crystals: 100
RANDOM 10 randomparameter: 25000
crystalparameter: 25000
LL[1-3]: 3.20 -7.30 5.10

RESULTS:
steps/fault

110	0.0080	79	0.0083	116	0.0084	101	0.0012
96	0.0030	101	0.0067	115	0.0016	112	0.0022
99	0.0173	87	0.0095	91	0.0097	80	0.0096
104	0.0014	110	0.0050	109	0.0029	109	0.0029
107	0.0054	76	0.0010	106	0.0001	104	0.0018
105	0.0010	79	0.0109	103	0.0049	100	0.0019
110	0.0043	89	0.0139	83	0.0058	101	0.0062
95	0.0134	100	0.0057	81	0.0139	110	0.0059
97	0.0019	81	0.0140	89	0.0023	109	0.0064
84	0.0026	104	0.0033	100	0.0034	110	0.0042
116	0.0028	127	0.0021	81	0.0141	110	0.0027
80	0.0071	107	0.0064	103	0.0005	117	0.0049
113	0.0105	105	0.0049	123	0.0010	82	0.0034
107	0.0011	79	0.0021	114	0.0043	91	0.0102
82	0.0084	114	0.0027	109	0.0004	83	0.0097
80	0.0095	107	0.0109	111	0.0005	89	0.0077
104	0.0021	112	0.0009	95	0.0025	102	0.0010
111	0.0025	111	0.0018	84	0.0133	110	0.0048
115	0.0028	112	0.0034	118	0.0000	109	0.0046
105	0.0007	84	0.0006	93	0.0045	97	0.0044
104	0.0062	76	0.0034	86	0.0038	106	0.0027
82	0.0005	105	0.0015	107	0.0004	105	0.0008
108	0.0028	112	0.0044	83	0.0021	75	0.0117
108	0.0048	109	0.0030	111	0.0014	103	0.0022
90	0.0060	96	0.0170	88	0.0081	104	0.0030

```
begin comment: A.E.K. - ADM 28 - July 1965;
integer pagecounter, linecounter, problem no,day,month,year,drum,a,
        xxx,yyy,pllll,llll,plabc;
real deltax,deltay;

procedure head; CR(100);

procedure outer;CR(1);

procedure stop;
begin integer a;
writetext(‡<
stop‡));
a := typechar;
if a ≥ 128 then a := a - 128;
if a = 50 then go to start;
if a = 53 then go to end of program;
end of stop;

procedure CR(a);
value a; integer a;

begin
if linecounter - 6 < a then a := linecounter + 2;
linecounter := linecounter - a;
for a := a - 1 step -1 until 0 do outchar(64);

if linecounter < 0 then begin
pagecounter := pagecounter + 1;
linecounter := linecounter + 64;
if pagecounter > 1 then
begin outsp(32); output(‡-ddd‡, -pagecounter, outtext(‡<-‡)) end;
outtext(‡<

A.E.K. - Program nr. 386  - Opgave nr.‡);
output(‡ndd‡, problem no);
outtext (‡< - ‡);
output(‡nd‡, day, outtext (‡<.‡), month); outtext(‡<.‡); output(‡nddd‡,year);

comment NOW ONE LINE TO BE PRINTED IN EACH HEADING CAN BE WRITTEN;

outtext(‡<


RCLTEX 1


‡)
end of linecounter <0
end of CR;

procedure plotline(x0,y0,x1,y1);
value x0, y0,x1,y1; real x0,y0,x1,y1;
begin integer xx,yy,tr; boolean hop; boolean array B[1:1111];
comment SA 74;
tr := drumplace; drumplace := pllll;
from drum(B); drumplace := tr;
xx := x0/deltax-xxx; yy := y0/deltay-yyy;
gierproc(B[2],hop,xx,yy);
xxx := xxx + xx; yyy := yyy + yy;
xx := x1/deltax - xxx; yy := y1/deltay - yyy;
gier(hop);
xxx := xxx + xx; yyy := yyy + yy
end plotline;
```

```
real procedure plotsymbol(t,x,y,h);
value t,x,y,h; integer t; real x,y,h;
begin integer xx,yy,n1,n2,hh,i,x1,y1,senk; boolean hop, AA;
boolean array A[0:1],B[1:1111];
comment SA 77/1;
hh := h x 20;
xx := drumplace; drumplace := pl111;
from drum(B); drumplace := plabc - t x 2;
from drum(A); drumplace := xx;
AA := A[0];
x1 := split(AA,0,0,senk,1,4,n1,5,9,y1) : 6;
y1 := (y1-x1x6) x hh; x1 := x1 x hh;
xx := x1 + x/deltax - xxx; yy := y1 + y/deltay - yyy-senkxhhx2;
xxx := x/deltax; yyy := y/deltay;
gierproc(B[2],hop,xx,yy);
n2 := (n1-8) x 5; n1 := if n1 ≤ 7 then n1 x 5 else 35;

for i := 10 step 5 until n1, 0 step 5 until n2 do
begin
      if i = 0 then AA := A[1];
      xx := split(AA,i,i+4,yy) : 6;
      yy := (yy-xxx6) x hh-y1;
      xx := xx x hh - x1;
      gier(hop);
      x1 := x1 + xx; y1 := y1 + yy end i;
xxx := xxx + x1; yyy := yyy + y1-senkxhhx2;
plotsymbol := x + split(AA,35,39,yy)/.3x h x deltax
end plotsymbol;


plabc :=plam12;1111:=40;pl111:=plabc-160;
xxx:=yyy:=0;deltax:=deltay:=1;
drum:=drumplace;
linecounter:=0;
writetext(‡<plotter‡);typechar;


start:
drumplace := drum;
pagecounter := 0;
problem no := inone;
if problem no  < 0 then go to end of program;

input(day,month,year);
for a := 1 step 1 until 30 do outchar(112);
head;
if year < 1968 ∨ year > 1980 then begin
writetext(‡<
aar‡); write(‡-nddd_b-dd‡,year); writetext(‡< i opg.‡); write(‡ndd‡,problem no);
stop;
end of test;
comment NOW COMES THE PROGRAM;



comment ROLTEX 1, P-586.  AEK 26 okt. 1966;
begin
real K,Ks,L,R,step,sq;
integer i,j,indicator,slipnumber,RANDOM,randomparameter,k,l,m,counter,
        derum,cx,cy;
real array A[1:3,1:3];
```

```
real procedure random(A,B,Y);
real A,B;
integer Y;
begin
integer C; own integer x;
if Y ≠ 0 then x := Y;
         C := x × 125;
         x := C-2796203 × entier(C/2796203);
         random := x × (B-A)/2796203+A;
end random;
procedure SLIP;

begin
real red,length,thickness,transdim,SA,d1,d2,d3,sumf,sumR,xpole,ypole;
integer p,q,r,s,P,Q,PP,QQ,PQ,Rcounter, fault;
real array N,n,P,S[1:4,1:3],D,d[1:4,1:3,1:3],f,fr, x[1:3,1:3];
integer array rcounter,RC[1:3,1:3];
          A[3,1] := A[1,2] × A[2,3]-A[2,2] × A[1,3];
          A[3,2] := A[1,3 ] × A[2,1]-A[2,3] × A[1,1];
          A[3,3] := A[1,1] × A[2,2]-A[2,1] × A[1,2];
if abs(1-sqrt(A[3,1] ↑ 2 + A[3,2] ↑ 2 + A[3,3] ↑ 2)) > 3₁₀-4  then
begin
outtext(‹<z-fault for(k,l)=›); output(‹ndd›,k);outtext(‹<,›);output(‹nd›,l);
outcr;end;
for q := 1 step 1 until 3 do
for r := 1 step 1 until 3 do
begin rcounter[q,r] := 0; fr[q,r] := f[q,r]:= 0;
      RC[q,r] := 2;
end;
      Rcounter := PQ := fault := 0; counter := -1;
      red := sumf := sumR := 0;
      length := thickness := transdim := 1;
comment note S1;

      N[1,1] :=N[2,1] :=N[3,1] :=N[4,1] :=N[1,2] :=N[2,2] :=N[1,3] :=N[3,3] :=sqrt(1/3);
      N[3,2] :=N[4,2] :=N[2,3] :=N[4,3] :=-N[1,1];
      D[1,1,1] :=D[1,2,1] :=D[2,1,1] :=D[2,2,1] :=D[3,1,1] :=D[3,2,1] :=D[4,1,1] :=
           D[4,2,1] :=D[1,3,2] :=D[2,3,2] :=D[3,1,2] :=D[3,3,2] :=D[4,1,2] :=D[4,3,2] :=
           D[2,2,3] :=D[2,3,3] :=D[3,3,3] :=D[4,2,3] :=sqrt(1/2);
      D[1,1,2] :=D[2,1,2] :=D[1,2,3] :=D[1,3,3] :=D[3,2,3] :=D[4,3,3] :=-D[1,1,1];
      D[1,3,1] :=D[2,3,1] :=D[3,3,1] :=D[4,3,1] :=D[1,2,2] :=D[2,2,2] :=D[3,2,2] :=
           D[4,2,2] :=D[1,1,3] :=D[2,1,3] :=D[3,1,3] :=D[4,1,3] := 0;
comment end note S1;
comment note S2;
new slip event:
      counter := counter +1;
for p := 1 step 1 until 4 do
for q := 1 step 1 until 3 do
for r := 1 step 1 until 3 do
begin
      n[p,r] :=d[p,q,r] :=0;
for s := 1 step 1 until 3 do
begin
      n[p,r] :=n[p,r] + N[p,s] × A[s,r];
      d[p,q,r] :=d[p,q,r] + D[p,q,s] × A[s,r];
end;
end note S2;
```

```
if abs(indicator) = 1 then go to plot;
if indicator = 3 then go to plot;
if red > R/100 then go to plot;
if counter = slipnumber then
begin
if slipnumber > 0 then go to plot;
end;
comment note S3;
go on:
        SA := 0;
for p := 1 step 1 until 4 do
begin
        F[p,1] := 0;
        F[p,2] := n[p,2];
        F[p,3] := -n[p,3];
for q := 1 step 1 until 3 do
for r := 1 step 1 until 3 do
begin
        F[p,q] := F[p,q] + n[p,r] × (f[r,q] + fr[r,q]);
if p = 4 then
begin
        rcounter[q,r] := rcounter[q,r] + 1;
if rcounter[q,r] = RC[q,r] then
begin
if q > r then
begin
        fr[q,r] := fr[r,q];
        RC[q,r] := RC[r,q];
        rcounter[q,r] := 0;
end
else
begin rcounter[q,r] := 0;
        RC[q,r] := random(1,RANDOM,0);
if L > -0.5 then
begin
        fr[q,r] := L × random(-100,100,0)/50;
if q ≠ r then
        fr[q,r] := fr[q,r]/4;
end
else
begin
        fr[q,r] := 5 × sumf/(10 × counter + 1) × random(-100,100,0)/50;
if q ≠ r then
        fr[q,r] := fr[q,r]/4;
end; end; end;
end; end;

for q := 1 step 1 until 3 do
begin
        S[p,q] :=F[p,1] × d[p,q,1] + F[p,2] × d[p,q,2] + F[p,3] × d[p,q,3];
if abs(S[p,q]) > SA then
begin
        SA := abs(S[p,q]);
        P := p; Q := q;
if S[p,q] < 0 then
begin
        d1 := -d[p,q,1];
        d2 := -d[p,q,2];
        d3 := -d[p,q,3];
end
else
begin
        d1 := d[p,q,1];
        d2 := d[p,q,2];
        d3 := d[p,q,3];
```

```
end; end; end;
end note S3;
comment note S4;
       A[2,1] :=-step × n[P,2] × d1;
       A[2,2] := 1-step × n[P,2] × d2;
       A[2,3] :=-step × n[P,2] × d3;
       length := (1+step × n[P,2] × d2) × length;
       sq := A[2,1] ↑ 2 + A[2,2] ↑ 2 + A[2,3] ↑ 2;
for r := 1 step 1 until 3 do
       A[2,r] := A[2,r]/sqrt(sq);
       A[1,1] := 1-step × n[P,1] × d1;
       A[1,2] := step × n[P,2] × d1;
       A[1,3] := -step × n[P,1] × d3;
       transdim := (1+step × n[P,1] × d1) × transdim;
       sq := A[1,1] ↑ 2 + A[1,2] ↑ 2 + A[1,3] ↑ 2;
for r := 1 step 1 until 3 do
       A[1,r] := A[1,r]/sqrt(sq);
       A[3,1] := A[1,2] × A[2,3] -A[2,2] × A[1,3];
       A[3,2] := A[1,3] × A[2,1] -A[2,3] × A[1,1];
       A[3,3] := A[1,1] × A[2,2] -A[2,1] × A[1,2];
       thickness := 1/(length × transdim);
       red := 1-thickness;
for p := 1 step 1 until 4 do
for q := 1 step 1 until 3 do
for r := 1 step 1 until 3 do
begin
       N[p,r] := n[p,r];
       D[p,q,r] := d[p,q,r];
end;
comment end note S4;
comment note S5;
      f[1,1] := K × (1-transdim);
      f[2,2] := K × (1-1/transdim);
      f[2,1] := f[1,2] := f[1,2] - Ks × step × 0.5 × (n[P,1] × d2 + n [P,2] × d1);
      f[1,3] := f[3,1] := f[3,1] - Ks × step × 0.5 × (n[P,3] × d1 + n [P,1] × d3);
      f[2,3] := f[3,2] := f[3,2] - Ks × step × 0.5 × (n[P,3] × d2 + n [P,2] × d3);
if L < -0.5 then
begin
       sumf := sumf + abs(f[1,1]) + abs(f[2,2]);
if Ks > 0.1 then
sumf := (sumf + 4× (abs(f[1,2]) + abs(f[2,3]) + abs(f[3,1])))/2.5;
if counter < 1 then
begin
       PP := P; QQ := Q;
end;
if PP = P then
begin
if QQ = Q then
begin
       PQ := PQ + 1;
go to no change for PP and QQ;
end; end;
       Rcounter := Rcounter + 1;
       sumR := sumR + PQ;
       PQ := 1;
if counter > RANDOM then
       RANDOM := 2 × sumR/Rcounter;
       PP := P; QQ := Q;
no change for PP and QQ:
if PQ > RANDOM/2 then
       RANDOM := RANDOM + 2 × (PQ-RANDOM/2)/(Rcounter + 1);
end note S5;
go to new slip event;
comment note S6;
plot:
for p := 1 step 1 until 4 do
```

```
begin
        xpole := n[p,1]/(abs(n[p,3]) + 1);
        ypole := n[p,2]/(abs(n[p,3]) + 1);
if n[p,3] < 0 then
begin
        xpole := -xpole;
        ypole := -ypole;
end;
if indicator = 3 then
begin
if counter = 0 then begin cx:= 1243+990*xpole; cy:= 3690+990*ypole;
        plotsymbol(166,cx,cy,.2) end
else
begin
if counter = slipnumber then begin cx:= 1207+990*xpole; cy:= 3661+990*ypole;
        plotsymbol(130,cx,cy,.9) end
else
begin
if red  > R/100 then
begin
  cx:= 1207 + 990 * xpole; cy := 3661 + 990 * ypole;
  plotsymbol (130 , cx, cy, 0.9 );
end
else begin cx:=1238+990*xpole;cy:=3688+990*ypole;
        plotsymbol(130,cx,cy,.3); end; end;
end; end
else
begin
cx:=1238+990*xpole;cy:=3688+990*ypole;
      plotsymbol(130,cx,cy,.3);
if indicator < 0 then
begin
if indicator > -2.5 then
begin cx:=1238-990*xpole;cy:=3688+990*ypole;
        plotsymbol(130,cx,cy,.3);
cx:=1238+990*xpole;cy:=3688-990*ypole;
        plotsymbol(130,cx,cy,.3);
cx:=1238-990*xpole;cy:=3688-990*ypole;
        plotsymbol(130,cx,cy,.3);
end; end; end;
end of 111 pole figure;
for q := 1 step 1 until 3 do
begin
        x[1,q] := n[1,q] + n[4,q];
        x[2,q] := n[1,q] - n[3,q];
        x[3,q] := n[1,q] - n[2,q];
end;
for p := 1 step 1 until 3 do
begin
        sq := x[p,1] ↑ 2 + x[p,2] ↑ 2 + x[p,3] ↑ 2;
for q := 1 step 1 until 3 do
        x[p,q] := x[p,q]/sqrt(sq); if p = 3 then begin
if fault < 0.5 then
begin
sq := x[1,1] * x[2,1] + x[1,2] * x[2,2] + x[1,3] * x[2,3];
if abs(sq) < 0.0 02  then
sq := x[2,1] * x[3,1] + x[2,2] * x[3,2] +x[2,3] * x[3,3];
if abs(sq) < 0.0 02  then
sq := x[3,1] *x[1,1] + x[3,2] * x[1,2] + x[3,3] * x[1,3];
if abs(sq) > 0.0 02  then
beginfault := 1;
outtext(‹<<100>-fault for (k,l)=›);
output(‹ndd›,k); outtext(‹<,›);
output(‹nd›,1); outcr;
end;end; end;
```

```
        xpole := x[p,1]/(abs( x[p,3]) + 1);
        ypole := x[p,2]/(abs( x[p,3]) + 1);
if x[p,3] < 0 then
begin
        xpole := - xpole;
        ypole := - ypole;
end;
if indicator = 3 then
begin
if counter = 1 then begin
cx:=1243+990xxpole;cy:=1290+990xypole;
        plotsymbol(166,cx,cy,.2) end
else
begin
if counter = slipnumber then begin cx:=1207+990xxpole;cy:=1261+990xypole;
        plotsymbol(130,cx,cy,.9)
end
elsebegin
if red  ≥ R/100 then
begin
  cx := 1207 + 990 x xpole; cy:= 1261 + 990 x ypole;
  plotsymbol (130 , cx, cy, 0.9);
end
else
begin cx:=1238+990xxpole;cy:=1288+990xypole;
        plotsymbol(130,cx,cy,.3); end; end;
end; if p = 3 then begin
if slipnumber > 0.5 then
begin
if counter < slipnumber then go to go on;
end
else
begin
if red < R/100 then go to go on;
end; end;
end
else
begin cx:=1238+990xxpole;cy:=1288+990xypole;
        plotsymbol(130,cx,cy,.3);
if indicator < 0 then
begin
if indicator > -2.5 then
begin
cx:=1238-990xxpole;cy:=1288+990xypole;
        plotsymbol(130,cx,cy,.3);
cx:=1238+990xxpole;cy:=1288-990xypole;
        plotsymbol(130,cx,cy,.3);
cx:=1238-990xxpole;cy:=1288-990xypole;
        plotsymbol(130,cx,cy,.3);
end; end; end;
end of 100 pole figure;
end of SLIP;

procedure plottext(string,x,y,h,d);
value h,d,string;  real x,y,h,d;  boolean string;
begin boolean array b[0:39];
integer i,j,tp,t,case; real x0;
x0 := x;  case := 0;
if split(string,0,3,t) = 10 then begin i :=
for i := i-6 while split(string,i,i+5,t)
do if t = 63 then begin x := x0; y := y-d x h end else begin
if t = 58 then case := 0
else if t = 60 then
case := 128 else x:= plotsymbol(t+case,x,y,h) end end else
begin
tp := drumplace;
```

```
split(string,10,19,i,30,39,j);
drumplace := 40xj+39; from drum(b); j := 42;
for j := j - 6 while split(b[i],j,j+5,t) ≠ 10 do begin
if t = 63 then begin x := x0; y := y-dxh end else
begin if t = 58 then case := 0
else if t = 60 then case := 128 else
x := plotsymbol(t+case,x,y,h) end;
if j = 0 then begin j := 42;  i := i+1 end;
if i > 39 then begin i := 0; from drum(b) end
end;
drumplace := tp end;
end plottext;

procedure plotaxes(x0,y0,xmin,xmax,ymin,ymax,xmark,ymark);
value x0,y0,xmin,xmax,ymin,ymax,xmark,ymark;
real x0,y0,xmin,xmax,ymin,ymax,xmark,ymark;
begin real y1,y2;
integer xx,yy,tr,j;
boolean hop,plus,x;
boolean array B[1:1111];
procedure pil;
begin xx := -5; yy := -20; gier(hop);
xx := 10; yy := 0; gier(hop);
xx := - 5;yy := 20; gier(hop);
end;
tr := drumplace; drumplace := pl 111;
from drum(B); drumplace := tr;
plus := abs(yyyxdeltay -ymin) > abs(yyyxdeltay - ymax);
x := false;
for j := 1, 2 do begin
y1 := if plus then ymax else ymin;
xx := x0/deltax-xxx; yy := y1/deltay-yyy;
if x then gierproc(B[2],hop,yy,xx) else gierproc(B[2],hop,xx,yy);
xxx := xxx + xx; yyy := yyy + yy;
if plus then pil;

for y2 := y0 + entier((y1-y0)/ymark+(if plus then -.01 else 1.01)) x ymark
        while y2 < ymax ∧ y2 ≥ ymin
do begin
xx := 0; yy := y2/deltay - yyy;
gier(hop);
yyy := yyy + yy;
xx := 10; yy := 0; gier(hop); xx := -20; gier(hop); xx := 10; gier(hop);
y1 := y2; end;
xx := 0; yy:=(if plus then ymin else ymax)/deltay - yyy;
gier(hop); yyy:= yyy+yy;
if -, plus then pil;
x := plus := true; ymin := xmin; ymax := xmax;
y1 := ymark; ymark:= xmark; xmark:= y1;
y1 := y0; y0 := x0; x0 := y1; y1 := xxx; xxx := yyy; yyy := y1;
y1 := deltax; deltax := deltay; deltay := y1; end;
end;


procedure plotcopy(string,x,y,h,d);
value string,h,d; real x,y,h,d; boolean string;
begin
real x0;
integer case,t,u,a;
x0 := x; case := 0;
split(string,34,39,t); a := 28;
if t = 60 then begin
case := 128; split(string,28,33,t); t := t + case;
a := 22; end;
if t = 63 then t := 64;
```

```
split(string,s-6,u-6,u);
if u = 58 ∨ u = 59 then begin
case := (u-58) x 64;
split(string,s-6,u-6,u); end;
if u = 63 then u := 64;
u := u + case;
if u = 10 then t := ' else
for t := t while inchar # t do;
for t := inchar while t # u do
if t = 'x' ∨ t = 'X' then begin
x := x0; y := y-d x 1 end
elsebegin
x:=plotsymbol(t,x,y,1);
outchar(t); end;
end of plotcopy;


comment note 1;
outer;outer;outer;
outtext(‡<GENERAL VARIABLES‡});
outer;outer;
outtext(‡<plot sign: ‡}); cx:=2100;cy:=4700;
plotcopy(‡<[ ]‡},cx,cy,0.6,1.5);
input(i,j,indicator,slipnumber,step,K,L,R,RANDOM,randomparameter);
outtext(‡<     indicator: ‡});
output(‡n‡},indicator);
outtext(‡<     slip number: ‡});
output(‡nddd‡},slipnumber);outtext(‡<     shape force factor: ‡});
output(‡ndd.dd‡},K);outer;
outtext(‡<random force factor: ‡});
output(‡nd.dd‡},L);outtext(‡<     number of different x: ‡});
output(‡ndd‡},1);
outer;
outtext(‡<number of different y corresponding to each x: ‡});
output(‡ndd‡},');outtext(‡<     reduction in per cent: ‡});
output(‡nd.dd‡},R);outer;
outtext(‡<RANDOM: ‡});
output(‡ndd‡},RANDOM);
outtext(‡<     magnitude of steps: ‡});
output(‡n.ddd‡},step);
outtext(‡<     randomparameter: ‡});
output(‡ndddd‡},randomparameter);
outer; outer; outer;
outtext(‡<COORDINATE VARIABLES‡});
if indicator> 1.5 then begin

outer;outer;
outtext(‡<  xx        xy        xz        yy        yz      steps‡});
outer;
end
else begin outer;outer;
outtext(‡<  xx        xy        xz‡});
outer;end;

if K > 0 then
   Ks:= K/4
else begin K := -K; Ks :=  ;
end;


comment end note 1;
comment note 2;
cx:=2420; cy:=3690;
      plottext(‡<TLE‡},cx,cy,.4,2);
cx:=1000;cy:=4750;
      plottext(‡<111 FOR PROBLEM‡},cx,cy,.4,2);
cx:=1910;cy:=4650;
      plottext(‡<NUM‡},cx,cy,.4,2);
```

```
            plotaxes(1250,3700,150,2350,2600,4800,990,990);
cx:=1200;cy:=2350;
      plottext(‹<100 FOR PROBLEM›,cx,cy,.4,2);
cx:=1218;cy:=2470;
      plottext(‹<RUN›,cx,cy,.4,2);
cx:=2420;cy:=1280;
      plottext(‹<TUB›,cx,cy,.4,2);
      plotaxes(1250,1570,150,2350,200,2400,990,990);
comment end note 2;
      sq := random(0,10000,randomparameter);
comment note 3;
if indicator < 1.5 then
for k := 1 step 1 until 1 do
begin
      input(A[1,1],A[1,2],A[1,3]);
output(‹-nd.dd›,A[1,1]); outsp(4);
output(‹-nd.dd›,A[1,2]); outsp(4);
output(‹-nd.dd›,A[1,3]); outcr;
      sq := A[1,1] ↑ 2 + A[1,2] ↑ 2 + A[1,3] ↑ 2;
for I := 1 step 1 until 3 do
      A[1,I] := A[1,I]/sqrt(sq);
if abs(A[1,3]) > 0.9999 then
begin
      A[2,3] := 0;
for J := 1 step 1 until J do
begin
      A[2,1] := cos(1/2 × 3.14159 × 1/J);
      A[2,2] := sin(1/2 × 3.14159 × 1/J);
      SLIP;
end; end
else for J := 1 step 1 until J do
begin
      A[2,3] := sqrt(A[1,1] ↑ 2 + A[1,2] ↑ 2) × sin(3.14159/(2 × J) × (1-1/4+(-1)↑K/5));
     A[2,2] := -A[1,2] × A[1,3] × A[2,3]/(A[1,1] ↑ 2 + A[1,2] ↑ 2) + sqrt
          ((A[1,2] × A[1,3] × A[2,3] × 2) ↑ 2 -4 × (A[1,1] ↑ 2 + A[1,2] ↑ 2) × ((A[1,1]
          × A[2,3]) ↑ 2 + (A[1,3] × A[2,3])↑ 2 -A[1,1] ↑ 2))/(2 × (A[1,1] ↑ 2 + A[1,2]
          ↑ 2));
      A[2,1] := sqrt(1-(A[2,2] ↑ 2 + A[2,3] ↑ 2));
if abs(A[1,1]  × A[2,1] + A[1,2] × A[2,2] + A[1,3] × A[2,3]) > 3₁₀-4  then
      A[2,1] := -A[2,1];
if abs(A[1,1] × A[2,1] + A[1,2] × A[2,2] + A[1,3] × A[2,3]) > 3₁₀-4  then
begin
outtext(‹<y-fault for(k,1) =›);
output(‹ndd›,k); outtext(‹<,›);
output(‹ndd›,1); outcr;
end;
      SLIP;
end;
end note 3
comment note 4
else
for k := 1 step 1 until 1 do
begin
      input(A[1,1], A[1,2], A[1,3], A[2,2],A[2,3]);
output(‹-nd.dd›,A[1,1]); outsp(4);
output(‹-nd.dd›,A[1,2]); outsp(4);
output(‹-nd.dd›,A[1,3]); outsp(4);
output(‹-nd.dd›,A[2,2]); outsp(4);
                  output(‹-nd.dd›,A[2,3]); outsp(4);
      sq := A[1,1] ↑ 2 + A[1,2] ↑ 2 + A[1,3] ↑ 2;
for I := 1 step 1 until 3 do
      A[1,I] := A[1,I]/sqrt(sq);
if abs (A[1,1])  > 0.001 then
      A[2,1] := -(A[1,2] × A[2,2] + A[1,3] × A[2,3])/A[1,1]
```

```
else
begin
A[2,1]:=0.1;
if aba (A[1,2] x A[2,2] + A[1,3] x A[2,3]) > 3E-4
then
begin
outtext (<<yy-yz/x-fault for K=>);
output (<n>,K); outcr;
end; end;

      sq := A[2,1] ↑ 2 + A[2,2] ↑ 2 + A[2,3] ↑ 2;
for I := 1 step 1 until 3 do
      A[2,I] := A[2,I]/sqrt(sq);
     SLIP;
output(<nddd>,counter); outcr;
end note 4; plotline (0,,500,0,5500);
xxx:= yyy:= 0;
end of ROLTEX;


comment THE FOLLOWING WILL BE INSERTED AFTER THE PROGRAM;
go to start;

end of program:
for a:=1 step 1 until linecounter do outcr;
outsum; for a := 1 step 1 until 60 do outchar(112);
end;
```

```
begin comment: A.E.K. - ADM 2B - July 1965;
integer pagecounter, linecounter, problem no,day,month,year,drum,a,
        xxx,yyy,pl111,l111,plabc;
real deltax,deltay;

procedure head; CR(100);

procedure outer;CR(1);

procedure stop;
begin integer a;
writetext(‡<
stop‡);
a := typechar;
if a > 128 then a := a - 128;
if a = 50 then go to start;
if a = 53 then go to end of program;
end of stop;

procedure CR(a);
value a; integer a;

begin
if linecounter - 6 < a then a := linecounter + 2;
linecounter := linecounter - a;
for a := a - 1 step -1 until 0 do outchar(64);

if linecounter < 0 then begin
pagecounter := pagecounter + 1;
linecounter := linecounter + 64;
if pagecounter > 1 then
begin outsp(32); output(‡-ddd‡, -pagecounter, outtext(‡<-‡)) end;
outtext(‡<

A.E.K. - Program nr. 464 - Opgave nr.‡);
output(‡ndd‡, problem no);
outtext (‡< - ‡);
output(‡nd‡, day, outtext (‡<.‡), month); outtext(‡<.‡); output(‡nddd‡,year);
comment NOW ONE LINE TO BE PRINTED IN EACH HEADING CAN BE WRITTEN;

outtext(‡<
HORS dUEUVRE


‡)
end of linecounter <0
end of CR;

procedure plotline(x0,y0,x1,y1);
value x0, y0,x1,y1; real x0,y0,x1,y1;
begin integer xx,yy,tr; boolean hop; boolean array B[1:1111];
comment SA 74;
tr := drumplace; drumplace := pl111;
from drum(B); drumplace := tr;
xx := x0/deltax-xxx; yy := y0/deltay-yyy;
gierproc(B[2],hop,xx,yy);
xxx := xxx + xx; yyy := yyy + yy;
xx := x1/deltax - xxx; yy := y1/deltay - yyy;
gier(hop);
xxx := xxx + xx; yyy := yyy + yy
end plotline;

real procedure plotsymbol(t,x,y,h);
value t,x,y,h; integer t; real x,y,h;
```

```
begin integer xx,yy,n1,n2,hh,i,x1,y1,sænk; boolean hop, AA;
boolean array A[0:1],B[1:1111];
comment SA ??/1;
hh := F x 20;
xx := drumplace; drumplace := pl111;
from drum(B); drumplace := plabc - t x 2;
from drum(A); drumplace := xx;
AA := A[0];
x1 := split(AA,0,0,sænk,1,4,n1,5,9,y1) : 6;
y1 := (y1-x1x6) x hh; x1 := x1 x hh;
xx := x1 + x/deltax - xxx; yy := y1 + y/deltay - yyy-sænkxhhx2;
xxx := x/deltax; yyy := y/deltay;
gierproc(B[2],hop,xx,yy);
n2 := (n1-8) x 5; n1 := if n1 ≤ 7 then n1 x 5 else 35;

for i := 1 step 5 until n1, 0 step 5 until n2 do
begin
      if i = 0 then AA := A[1];
      xx := split(AA,i,i+4,yy) : 6;
      yy := (yy-xxx6) x hh-y1;
      xx := xx x hh - x1;
      gier(hop);
      x1 := x1 + xx; y1 := y1 + yy end i;
xxx := xxx + x1; yyy := yyy + y1-sænkxhhx2;
plotsymbol := x + split(AA,35,39,yy)/.3x h x deltax
end plotsymbol;


plabc :=plsm12;l111:=40;pl111:=plabc-160;
xxx:=yyy:=0;deltax:=deltay:=1;
drum:=drumplace;
linecounter:=0;
writetext(‹<plotter›);typechar;
for a:=1 step 1 until 30 do outchar(112);outclear;


start:
drumplace := drum;
pagecounter := 0;
problem no := inone;
if problem no < 0 then go to end of program;

input(day,month,year);
for a := 1 step 1 until 30 do outchar(112);
head;
if year < 1965 ∨ year > 1970 then begin
writetext(‹<
aar›); write(‹-nddd_B-dd›,year); writetext(‹< i opg.›); write(‹ndd›,problem no);
stop;
end of test;
comment NOW COMES THE PROGRAM;

comment HORS dŒUVRE, P-464 A.E.K. 1 december 1967;

begin
real K, Ks, R, step, sq, u, red, length,
     thickness, transdim, SA, d1, d2, d3, sumf, sum1, sum2, sum3,
     cx, cy, fault;
integer i, STOP, crystalparameter, k, counter,
     sumcounter, p, q, r, s, t,M, P, Q;
real array A, f [1:3,1:3], N,n,F,S[1:4,1:3],
     D, d[1:4,1:3,1:3], LL [0:3],
     RED,SUMF,SUM1, SUM2, SUM3, ZERO[1:11];
```

```
real  procedure randomcryst (A,B,Y);
real A,B;
integer Y;
begin
integerC; own integer x;
if Y ≠ 0 then x:= Y;
      C:= x × 125;
      x:= C-2796203 × entier (C/2796203);
      randomcryst:= x × (B-A)/2796203+A;
end randomcryst;
real

procedure plottext(string,x,y,h,d);
value h,d,string;  real x,y,h,d;  boolean string;
begin boolean array b[0:39];
integer i,j,tp,t,case; real x0;
x0 := x;  case := 0;
if split(string,0,3,t) = 10 then begin i := 40;
for i := i-6 while split(string,i,i+5,t)≠10
do if t = 63 then begin x := x0; y := y-d × h end else begin
if t = 58 then case := 0
else if t = 60 then
case := 128 else x:= plotsymbol(t+case,x,y,h) end end else
begin
tp := drumplace;
split(string,10,19,1,30,39,j);
drumplace := 40×j+39; from drum(b); j := 42;
for j := j - 6 while split(b[i],j,j+5,t) ≠ 10 do begin
if t = 63 then begin x := x0; y := y-d×h end else
begin if t = 58 then case := 0
else if t = 60 then case := 128 else
x := plotsymbol(t+case,x,y,h) end;

if j = 0 then begin j := 42;  i := i+1 end;
if i > 39 then begin i := 0; from drum(b) end
end;
drumplace := tp end;
plottext:=x;
end plottext;

procedure plotaxes(x0,y0,xmin,xmax,ymin,ymax,xmark,ymark);
value x0,y0,xmin,xmax,ymin,ymax,xmark,ymark;
real x0,y0,xmin,xmax,ymin,ymax,xmark,ymark;
begin real y1,y2;
Integer xx,yy,tr,j;
boolean hop,plus,x;
boolean array B[1:1111];
procedure pil;
begin xx := -5; yy := -20; gier(hop);
xx := 10; yy := 0; gier(hop);
xx := - 5;yy := 20; gier(hop);
end;
tr := drumplace; drumplace := pl 111;
from drum(B); drumplace := tr;
plus := abs(yyy×deltay -ymin) > abs(yyy×deltay - ymax);
x := false;
for j := 1, 2 do begin
y1 := if plus then ymax else ymin;
xx := x0/deltax-xxx; yy := y1/deltay-yyy;
if x then gierproc(B[2],hop,yy,xx) else gierproc(B[2],hop,xx,yy);
xxx := xxx + xx; yyy := yyy + yy;
if plus then pil;
```

```
for y2 := y0 + entier((y1-y0)/ymark+(if plus then -.01 else 1.01)) x ymark
        while y2 < ymax ∧ y2 ≥ ymin
do begin
xx := 0; yy := y2/deltay - yyy;
gier(hop);
yyy := yyy + yy;
xx := 10; yy := 0; gier(hop); xx := -20; gier(hop); xx := 10; gier(hop);
y1 := y2; end;
xx := 0; yy:=(if plus then ymin else ymax)/deltay - yyy;
gier(hop); yyy:= yyy+yy;
if -, plus then pil;
x:= plus := true; ymin := xmin; ymax := xmax;
y1 := ymark; ymark:= xmark; xmark:= y1;
y1 := y0; y0 := x0; x0 := y1; y1 := xxx; xxx := yyy; yyy := y1;
y1 := deltax; deltax := deltay; deltay := y1; end;
end;



real procedure plotnumber(layout,tal,x,y,h);
value tal,x,y,h;
string layout;  real tal,x,y,h;

begin comment A.E.K.-SA 91, april 1967.Description in SA 91;
integer ans,int,magn,exp,ti,zero,spaces,forp,eftp,fn,n,fe,inexp;

procedure anslag(symbol); integer symbol;
begin
      if symbol=0 then x:=x+h else x:=plotsymbol(symbol,x,y,h);
      if ans>1 then ans:=ans/2;
      if spaces> ans then begin spaces:=spaces-ans;x:=x+h end
end anslag;

procedure plotint(int,length,n,fn,pointplace);
value   int,length,n,fn,pointplace;
integer int,length,n,fn,pointplace;
begin integer ciffer,q;boolean signif;

if fn=3 then begin x:=plotsymbol(if int<0 then 32 else 160,x,y,h);fn:=0 end

       else fn:= if int<0 then 32 else if fn=2 then 160 else fn×128;
if fn=128 then begin x:=x+h;fn:=0 end;
int:=abs(int); signif:=false;q:=ti↑length;

for length:= length-1 while length ≥ 0 do
begin
     q:=q/ti;ciffer:=int:q;int:=int-ciffer×q;
     if -,signif then signif:=ciffer≠0 ∨ length=pointplace-1+n;
     if   signif then begin if ciffer=0 then ciffer:=16;
                            if     fn≠1 then begin x:=plotsymbol(fn,x,y,h);fn:=0 end;
                            if length=pointplace-1 then anslag(59)
                      end;
     anslag(ciffer)
end whilestep;
if fn≠0 then x:=x+h
end plotint;

zero:= -split(layout,,,19,spaces,28,29,fn,34,34,n,24,27,forp,30,33,eftp,
38,39,fe,35,37,inexp,20,23, ero)+ forp+eftp;
if fn>1 ∧ eftp=0 then n:=1;

exp:=0;ti:=10;if tal<0 then begin fn:=fn-ti;tal:=-tal end;
ans:=-ti↑inexp;magn:=ti↑(forp-1);

comment venstreforskydning;
```

```
if fe=0 then exp:=-1
        else
for exp:=exp-1 while exp> ans ∧tal< magn do tal:= talxti;

comment højreforskydning;
magn:= if tal<1 then -1 else entier(ln(tal)/2.3025);
for exp:=exp+1 while magn>førp do begin tal:= tal/ti;magn:=magn-1end;

int:=entier(talxti∧eftp+.5);
if int=0 then zero:=exp:=0
         else
begin
      if int> ti∧(førp+eftp) then begin exp:=exp+1;int:=int/ti end;
      if zero<0 then
begin
      zero:=zero+1;
      if inexp ≠exp=0then magn:= 0
                     else begin magn:=entier(exp/zero+.99)xzero-exp;exp:=exp+magn end;
      zero:=zero+entier(ln(int)/2.3025)-førp-eftp-magn;
      if zero< 0 then int:=entier(int/ti∧magn+.5)
                 else
      begin int:=entier(int/ti∧(magn+zero)+.5);
            if eftp>zero then eftp:=eftp-zero
                         else
      begin if exp=0 then begin int:=intxti∧(zero-eftp);zero:=eftp;eftp:=0 end
                     else eftp:=eftp-zero;
            if eftp+zero≠0 then zero:=zero+1
      end
      end
end betydende cifre
end int≠0;

if fn<0 then begin int:=-int;fn:=fn+ti end;
ans:= 524 288;
plotint(int,førp+eftp,n,fn,eftp);
for zero:=zero-1 while zero>0 do anslag(0);

if exp=0 then for inexp:=inexp-1 while inexp> 0 do
              begin anslag(0);if inexp=0 then x:=x+h;
                    if fe≠0 then begin x:=x+h;fe:=0 end
              end
         else begin if exp> ti∧inexp then inexp:=entier(ln(exp)/2.3025)
                                     else
                    for inexp:=inexp-1 while fe+3∧ti∧inexp> abs(exp) do anslag(0);
                    x:=plotsymbol(155,x,y,h);
                    plotint(exp,inexp+1,10,fe,0)
              end;
plotnumber:=x
end plotnumber;

procedure plotgraph(x0,x1,x,y,dx);
value x0,x1; real x0,x1,x,y,dx;
begin integer xx,yy,tr; boolean hop; boolean array B[1:1111];
tr := drumplace; drumplace := pl111;
from drum(B); drumplace := tr;
x := x0; xx := x0/deltax-xxx; yy := y/deltay-yyy;
gierpro:(B[2],hop,xx,yy);
xxx := xxx + xx; yyy := yyy + yy;
for x := x0 step dx until x1 do
begin
xx := x/deltax-xxx; yy := y/deltay-yyy;
gier(hop);
xxx := xxx + xx; yyy := yyy + yy
end end plotgraph;
```

```
procedure POLY! (N,P,x,y,w,a,WEIGHING);
integer N,P;
boolean WEIGHING;
array x,y,w,a;

comment A.E.K. August 2nd. 1962. Least squares fit of a polynomial approximation
as described in SA - 1;

begin
integer j,k,n;
real alfa,beta,XPROD,YPROD,SQ,SQSUM,OLDSQSUM,h old;
array error, orpol,oldorpol[1:N], cora[-1:P], oldcora[ :P];

for n := 1 step 1 until N do
begin
error[n] := y[n];
orpol[n] := 0;
oldorpol[n] := 1
end of initial setting;

alfa := olda := cora[-1] := 0;
beta := OLDSQSUM := 1;

for k := 0 step 1 until P do
begin
XPROD := YPROD := SQSUM := 0;

for n := 1 step 1 until N do
begin
error [n] := error[n] - olda × orpol[n];
H := oldorpol[n] × beta;
oldorpol[n] := orpol[n];
R := orpol[n] := R - orpol[n] × (x[n] + alfa);
if WEIGHING then R := orpol[n] × w[n];
SQ := R × orpol[n];
SQSUM := SQSUM + SQ;
YPROD := YPROD + R × error[n];
XPROD := XPROD + SQ × x[n]
end for n;

a[k]:=olda:= YPROD/SQSUM;
oldcora [k]:= 0;
cora [k]:= 1;
if k>0 then

for j:= k-1 step -1 until 0 do
begin
H := beta × oldcora[j];
oldcora[j] := cora[j];
cora[j] := alfa × oldcora[j] + R + cora[j-1];
a[j] := a[j] + olda × cora[j]
end for j;

beta := -SQSUM/OLDSQSUM;
OLDSQSUM := SQSUM;
alfa := -XPROD/SQSUM
end for k;
end of POLY-1;

comment now coordinate system and input;
input(K, Ka, R, step, STOP, 1, crystalparameter);
CR(2); outtext(‹<CONSTANTS›); CR(2);
outtext(‹<K: ›); output(‹ndd.d›,K);
```

```
outtext(‡<     Ks: ‡); output(‡ndd.d‡,Ks);
outtext(‡<     R: ‡); output(‡nd.d‡,R); CR(1); outtext(‡<step: ‡);
output(‡n.ddd‡,step); outtext(‡<     STUP: ‡); output(‡ndd‡,STUP);
outtext(‡<     1: ‡); output(‡ndd‡,i);
CR(1); outtext(‡<crystalparameter: ‡);
output(‡ndddddd‡,crystalparameter);
CR(3);  outtext(‡<steps/fault‡); CR(2);
plotaxes (600, 800, 400, 2300, 300, 3300, 1500, 1000);
cx:= 100; cy:= 2500;
cx:= plottext (‡<HURS dUEUVRE - PROBLEM ‡, cx, cy, 0.4, 2);
plotnumber (‡ndddd‡,problem no, cx, cy, 0.4);
     t:= 0;
     sq:= randomcryst(0,100,crystalparameter);
for M:= 1 step 1 until 11 do
     SUMF[M]:=SUM1[M]:= SUM2[M]:= SUM3[M]:= 0;

comment  now the crystals one by one;
for k:= 1 step 1 until 1 do
begin
A[1,3]:= randomcryst(0,1,0);
if A[1,3] < 0.99999 then
begin
     u:= randomcryst (0,1,0);
     u:= randomcryst (0,1.57 796,0);
     A[1,1]:= cos (u) × sqrt (1-A[1,3] ↑2);
     A[1,2]:= sin (u) × sqrt (1-A[1,3] ↑2);
end else
begin
      A[1,1]:= A[1,2]:= 0;
      A[1,3]:= 1;
end;
     u:= randomcryst (0,1,0);
     u:= randomcryst (0,3.141593, 0);
if A[1,2] ↑ 2 + A[1,3] ↑ 2 < 0.99998 then
begin
     A[2,3]:= sin (u) × sqrt (A[1,1] ↑ 2 + A[1,2] ↑ 2);
     sq:= A[1,2] × A[1,3]/sqrt ((A[1,1] ↑2 +A[1,2] ↑2) × (A[1,1] ↑ 2 + A[1,3] ↑ 2));
     u:= u - arctan (sqrt(1-s↑2)/s);
     A[2,2]:= - sin (u) × sqrt (A[1,1] ↑ 2 +A[1,3] ↑ 2);
     A[2,1]:=- (A[1,2] × A[2,2] + A[1,3] × A[2,3])/A[1,1];
end else
begin
     A[1,1]:= 0;
     sq:= sqrt (A[1,2] ↑ 2 + A[1,3] ↑ 2);
     A[1,2]:= A[1,2]/sq;
     A[1,3]:= A[1,3]/sq;
     A[2,1]:= sin (u);
     A[2,2]:= cos (u) × A[1,3];
     A[2,3]:= -cos (u) × A[1,2];
end;
if abs (1-sqrt (A[2,1] ↑2 + A[2,2] ↑ 2 + A[2,3] ↑ 2)) > 3ₘ-4 then
outtext (‡<-y   ‡);
A[3,1]:= A[1,2]×A[2,3]-A[2,2]×A[1,3];
A[3,2]:= A[1,3]×A[2,1]-A[2,3]×A[1,1];
A[3,3]:= A[1,1]×A[2,2]-A[2,1]×A[1,2];
if abs (sqrt(A[3,1]↑2+A[3,2]↑2+A[3,3]↑2)-1) > 3ₘ-4 then
outtext (‡<-z   ‡);

comment coordinates being generated now slip;
for q:= 1 step 1 until 3 do
for r:= 1 step 1 until 3 do
        f[q,r]:= 0;
        counter:= sumcounter:= -1;
        red:= sumf:= fault:= sum1:= sum2:= sum3:= 0;
         length:= thickness:= transdim:= 1;
        RED[2]:= 0.1;
```

```
for M:= 3 step 1 until 11 do
        RED[M]:= RED[M-1]+0.1;

      N[1,1] :=N[2,1] :=N[3,1] :=N[4,1] :=N[1,2] :=N[2,2] :=N[1,3] :=N[3,3] :=sqrt(1/3);
      N[3,2] :=N[4,2] :=N[2,3] :=N[4,3] :=-N[1,1];
      D[1,1,1] :=D[1,2,1] :=D[2,1,1] :=D[2,2,1] :=D[3,1,1] :=D[3,2,1] :=D[4,1,1] :=
           D[4,2,1] :=D[1,3,2] :=D[2,3,2] :=D[3,1,2] :=D[3,3,2] :=D[4,1,2] :=D[4,3,2] :=
           D[2,2,3] :=D[2,3,3] :=D[3,3,3] :=D[4,2,3] :=sqrt(1/2);
      D[1,1,2] :=D[2,1,2] :=D[1,2,3] :=D[1,3,3] :=D[3,2,3] :=D[4,3,3] :=-D[1,1,1];
      D[1,3,1] :=D[2,3,1] :=D[3,3,1] :=D[4,3,1] :=D[1,2,2] :=D[2,2,2] :=D[3,2,2] :=
           D[4,2,2] :=D[1,1,3] :=D[2,1,3] :=D[3,1,3] :=D[4,1,3] := 0;
new slip event:
           counter:= counter+1;
           sumcounter:= sumcounter+1;
for p:= 1 step 1 until 4 do
for q:= 1 step 1 until 3 do
for r:= 1 step 1 until 3 do
begin
        n[p,r]:= d[p,q,r]:= 0;
for s:= 1 step 1 until 3 do
begin
        n[p,r]:= n[p,r]+N[p,s]×A[s,r];
        d[p,q,r]:=d[p,q,r]+D[p,q,s]×A[s,r];
end; end;
if red > R/100 ∨ counter ≥ STOP then
go to finish;

       SA := 0;
for p := 1 step 1 until 4 do
begin
       F[p,1] := 0;
       F[p,2] := n[p,2];
       F[p,3] := -n[p,3];
for q := 1 step 1 until 3 do
for r := 1 step 1 until 3 do
           F[p,q]:= F[p,q]+n[p,r]×r[r,q];
for q:= 1 step 1 until 3 do
begin
S[p,q]:= F[p,1]×d[p,q,1]+F[p,2]×d[p,q,2]+F[p,3]×d[p,q,3];
if abs(S[p,q]) > SA then
begin
       SA := abs(S[p,q]);
       P := p; Q := q;
if S[p,q] < 0 then
begin
       d1 := -d[p,q,1];
       d2 := -d[p,q,2];
       d3 := -d[p,q,3];
end
else
begin
       d1 := d[p,q,1];
       d2 := d[p,q,2];
       d3 := d[p,q,3];
end; end; end;
end;
       A[2,1] :=-step × n[P,2] × d1;
       A[2,2] := 1-step × n[P,2] × d2;
       A[2,3] :=-step × n[P,2] × d3;
       length := (1+step × n[P,2] × d2) × length;
       sq := sqrt( A[2,1] ↑ 2 + A[2,2] ↑ 2 + A[2,3] ↑ 2);
for r := 1 step 1 until 3 do
       A[2,r] := A[2,r]/sq;
       A[1,1] := 1-step × n[P,1] × d1;
       A[1,2] := step × n[P,2] × d1;
       A[1,3] := -step × n[P,1] × d3;
```

```
      transdim := (1+step × n[P,1] × d1) × transdim;
      sq := sqrt(A[1,1] ↑ 2 + A[1,2] ↑ 2 + A[1,3] ↑ 2);
for r := 1 step 1 until 3 do
      A[1,r] := A[1,r]/sq;
      A[3,1] := A[1,2] × A[2,3] -A[2,2] × A[1,3];
      A[3,2] := A[1,3] × A[2,1] -A[2,3] × A[1,1];
      A[3,3] := A[1,1] × A[2,2] -A[2,1] × A[1,2];
      thickness := 1/(length × transdim);
      red := 1-thickness;
for p := 1 step 1 until 4 do
for q := 1 step 1 until 3 do
begin
   N[p,q]:=n[p,q];
for r := 1 step 1 until 3 do
      D[p,q,r] := d[p,q,r];
end;
      f[1,1] := K × (1-transdim);
      f[2,2] := K × (1-1/transdim);
      f[2,1] := f[1,2] := f[1,2] - Ks × step × 0.5 × (n[P,1] × d2 + n [P,2] × d1);
      f[1,3] := f[3,1] := f[3,1] - Ks × step × 0.5 × (n[P,3] × d1 + n [P,1] × d3);
      f[2,3] := f[3,2] := f[3,2] - Ks × step × 0.5 × (n[P,3] × d2 + n [P,2] × d3);
     sumf:= sumf  +((abs(f[1,1])+abs(f[2,2]))/2+Ks×(abs(f[1,2])+
     abs(f[2,3])+abs(f[3,1]))/(3×K))/(1+Ks/K);
sum1:= sum1+(abs(f[1,1])+abs(f[2,2]))/2;
sum2:= sum2+(abs(f[1,2])+abs(f[1,3]))/2;
sum3:= sum3+abs(f[2,3]);
for  M:= 2 step 1 until 11 do
if red ≥ RED[M] then
begin
     RED[M]:= 10;
     SUMF[M]:= SUMF[M]+sumf/sumcounter;
SUM1[M]:= SUM1[M]+sum1/sumcounter;
SUM2[M]:= SUM2[M]+sum2/sumcounter;
SUM3[M]:= SUM3[M]+sum3/sumcounter;
sumf:= sum1:= sum2:= sum3:= 0;
     sumcounter:= 0;
end;
go to new slip event;
finish:
for p:= 2 step 1 until 4 do
begin
     sq:= abs(^.3333-abs(n[1,1]×n[p,1]+n[1,2]×n[p,2]+n[1,3]×n[p,3]));
if sq > fault then
     fault:= sq;

end;
output(‹nddd›,counter); outsp(2);
output(‹n.dddd›,fault);
     t:= t+1;
if t > 3.5 then
begin
     t:= 0; CR(1);
end else
outsp(5);
end CRYSTAL;

comment now results;
     RED[1]:= 0;
     RED[2]:= 0.05;
     ZERO[1]:= 100;
     ZERO[2]:= 1;
for M:= 3 step 1 until 11 do
begin
     RED[M]:= RED[M-1]+0.1;
     ZERO[M]:= 1;
end;
CR(3); outtext(‹<RESULTS›); CR(2);
```

```
for M:= 2 step 1 until 11 do
if RED[M] < R/100 then
begin
     SUMP[M]:= SUMP[M]/1;
SUM1[M]:= SUM1[M]/1;
SUM2[M]:= SUM2[M]/1;
SUM3[M]:= SUM3[M]/1;
output(<n.ddd>,RED[M]); outsp(6);
output(<n.dd₈-d>,SUMP[M],outsp(6), SUM1[M],outsp(6),SUM2[M],outsp(6),
       SUM3[M]); CR(1);
cx:= 1500xRED[M]+58⁸; cy:= 1000xSUMP[M]+7⁸⁸;
plotsymbol(13⁻,cx,cy,.3);
     P:= M;
end;
     POLY1 (P, 3, RED, SUMP, ZERO, LL, true);
CR(2); outtext(<LL[0:3]: >);
output(<-n.ddd₈-d>, LL[0], outsp(5),
     LL[1], outsp(5), LL[2], outsp(5), LL[3]);
cx:= 6 0; sq:= 600+RED[P]x1500;
plotgraph (cx, sq, red, 600+(LL[0]+LL[1]x((red-600)/1500)+LL[2]x((red-600)
     /1500)↑2+LL[3]x((red-600)/1500)↑3)x1000,10);

plotline (0,2600, ,2600);
xxx:= yyy:= 0;
end HURS dŒUVRE;

comment THE FOLLOWING WILL BE INSERTED AFTER THE PROGRAM;
go to start;

end of program:
for a:=1 step 1 until linecounter do outcr;
outsum; for a := 1 step 1 until 60 do outchar(112);
end;
```

A.E.K. - Program nr. 464 - Opgave nr. 20 - 19. 1.1973
HORS dOEUVRE

CONSTANTS

K: 20.0 Ks: 20.0 R: 80.0
step: 0.050 STOP: 150 i: 20
crystalparameter: 50000

steps/fault

78	0.0031	79	0.0010	78	0.0080	113	0.0034
114	0.0002	103	0.0106	115	0.0034	87	0.0097
111	0.0093	98	0.0111	116	0.0011	79	0.0027
80	0.0060	79	0.0060	118	0.0005	116	0.0013
104	0.0115	97	0.0052	111	0.0009	88	0.0098

RESULTS

0.050	1.49_{10}-1	1.01_{10}-1	1.87_{10}-1	2.16_{10}-1
0.150	1.78_{10}-1	1.41_{10}-1	1.96_{10}-1	2.54_{10}-1
0.250	1.97_{10}-1	1.68_{10}-1	2.12_{10}-1	2.50_{10}-1
0.350	2.13_{10}-1	2.13_{10}-1	2.06_{10}-1	2.29_{10}-1
0.450	2.13_{10}-1	2.34_{10}-1	1.83_{10}-1	2.11_{10}-1
0.550	2.23_{10}-1	2.58_{10}-1	1.85_{10}-1	1.94_{10}-1
0.650	1.98_{10}-1	2.08_{10}-1	1.84_{10}-1	1.97_{10}-1
0.750	1.99_{10}-1	2.05_{10}-1	1.85_{10}-1	2.07_{10}-1

LL[0:3]: 5.334_{10}-4 1.665 -3.79i 2.587